EDITORIALE

ATTACCO A FACEBOOOK

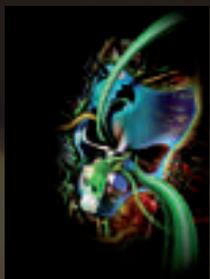
In questo numero 203 abbiamo un programma quantomai ricco che vede in bella evidenza un articolo che prende di mira le debolezze insite nel più popolare social network: Facebook, già al centro di molte polemiche, nel recente passato, per questioni di sicurezza e privacy.
Continua poi il nostro corso di C, che rappresenta ormai una certezza acquisita e che, dopo una prima parte teorica, si sta avventurando nei meandri più squisitamente tecnici. Ci terrà compagnia ancora per qualche numero.
La palma di articolo più stuzzicante spetta però a quello dedicato alla distanza di Levenshtein, ovvero a come implementare la nota funzione “forse cercavi…” del motore di ricerca Google.
Infine un assaggio di programmazione con SDK per iPad. Abbiamo creato un simulatore di gradimento che, peraltro, potete scaricare sul sito HJ nella sezione download.
Insomma anche in questo numero c’è molta carne al fuoco… buona lettura e buon hacking a tutti.

Altair

laboratorio@hackerjournal.it
Questo indirizzo è stato creato per inviare articoli, codici, spunti e idee. E’ quindi proprio una sorta di ”incubatore di idee”.

posta@hackerjournal.it
E’ l’account creato per l’omonima rubrica che è ricomparsa nelle pagine della rivista. A questo indirizzo dovete inviare tutte le mail che volete vengano pubblicate su HJ.

redazione@hackerjournal.it
Questo è l’indirizzo canonico. Quello con cui potete avere un filo diretto, sempre, con la redazione, per qualsiasi motivo che non rientri nelle due precedenti categorie di posta.

Sommario

Copertina:
Daniela Festa
ldfesta@libero.it

Anno 10 - N.203
Luglio 2010

Editore (sede legale)
WLF Publishing S.r.l.
Socio Unico Medi & Son S.r.l.
via Donatello 71 - 00196 Roma
Fax 063214606

Realizzazione editoriale
Progetti e promozioni Srl
redazione@progettiepromozioni.com

Printing
Grafiche Mazzucchelli S.p.a - Seriate (BG)

Distributore
M-DIS Distributore SPA
via Cazzaniga 2 - 20123 Milano

Hacker Journal
Pubblicazione quattordicinale registrata al Tribunale di Milano il 27/10/03 con il numero 601.
Una copia: 2,00 euro

Direttore Responsabile
Teresa Carsaniga
redazione@hackerjournal.it

DOVE VA A FINIRE L'EDITORIA?

Approfittiamo di questa terza pagina, dedicata agli argomenti di attualità, per trattare un tema che forse non è strettamente legato all'hacking ma che è certamente connesso con il mondo dell'editoria e, quindi, tocca, in qualche modo, anche Hacker Journal: il futuro della carta stampata.

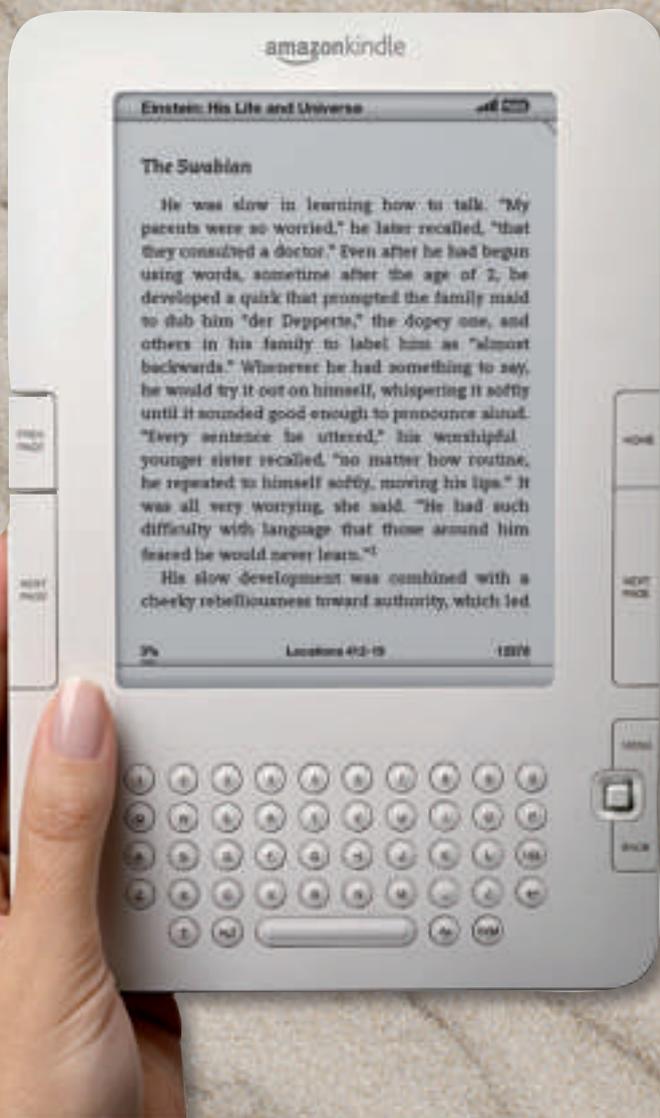
O meglio il non futuro. Infatti, l'avvento sul mercato di numerosi dispositivi che in qualche modo implementano la lettura dei testi digitali, iPad e Kindle su tutti ma i concorrenti si stanno facendo sotto. Sembra che l'editoria voglia convergere su questi nuovi dispositivi. C'è un entusiasmo contagioso, quasi irrefrenabile, per il momento assolutamente poco congruo ai numeri reali. Condenast ha provato a realizzare una versione speciale per iPad della sua rivista GQ. Risultato? 360 copie vendute, più o meno... Ovvero nulla.

Certo siamo agli albori. C'è chi è pronto a giurare che il futuro è questo, però qualche dubbio rimane.

Non è difficile credere che l'entusiasmo degli editori per questi nuovi terminali sia anche generato da una situazione contingente: le riviste vendono sempre meno e i costi di edicola, stampa e distribuzione si portano via più del 50 per cento dei guadagni. Rendere disponibile la propria rivista su un dispositivo elettronico significa annullare questi costi, incrementare quindi i guadagni (forse), sicuramente avere dei punti di pareggio molto più bassi e abbordabili.

Insomma l'editoria sta ripiegando su se stessa, preoccupandosi di spendere meno e limitare i costi aspettando segnali dal futuro.

E chissà che l'iPad, il Kindle o chi per essi non siano, oltre che una scelta conveniente sul breve, una soluzione lungimirante sul lungo periodo.

Solo il tempo ci dirà chi ha ragione. Il rischio a parere di molti è che nel giro di pochi anni la cara vecchia edicola diventi solo un ricordo, magari riconvertita in attività più remunerative.

PIÙ ARRESTI NEL DIGITALE

I primi quattro mesi del 2010 hanno fatto registrare un deciso incremento degli arresti e dei procedimenti giudiziari nei confronti della criminalità digitale in diversi Paesi. E' quanto emerge dal rapporto sulla sicurezza di F-Secure relativo alla prima parte dell'anno.
Nato come hobby, intorno al 2003 il malware è diventato un vero e proprio business controllato da criminali e capace di generare enormi profitti. Tuttavia, la trasformazione del malware da "passatempo" online in attività criminale per diversi anni non ha portato all'arresto degli autori degli attacchi. Nei rari casi in cui queste persone sono state fermate e processate, le sentenze non sono state quasi mai di condanna.
"Le aziende che si occupano di sicurezza informatica non sono forze di polizia, ma segnalano costantemente alle autorità il materiale scoperto nel corso delle indagini sul cyber crime", ha dichiarato Mikko Hyppönen, Responsabile dei Laboratori di Ricerca di F-Secure.
"Siamo soddisfatti di vedere che il nostro contributo sia utile alle autorità giudiziarie e speriamo che i recenti arresti e le condanne rappresentino un reale segnale di cambiamento nel modo di affrontare il crimine online".
E' dello scorso marzo la condanna esemplare a 20 anni di carcere di Alfredo Gonzales, a capo di una banda di criminali informatici colpevole di essersi impossessata dei dati di decine di milioni di carte di credito dalla catena di grandi magazzini americana TJ Maxx e da altri punti vendita negli Stati Uniti. Si tratta della sentenza più dura mai pronunciata per un caso di crimine informatico.
Gonzales e i membri della sua banda si sono infiltrati nei sistemi di autenticazione dei registri di cassa dei rivenditori attraverso una connessione wi-fi. E' stato necessario ristampare milioni di carte di credito.
Tra gli altri casi notevoli del 2010, la condanna a 5 anni di carcere da scontare nel Regno Unito a carico di Renu Subramaniam, alias JiLsi, uno dei criminali informatici del forum di hacker DarkMarket.
In Estonia, l'autore della famiglia di virus Allaple, Artur Boiko, è stato condannato a 2 anni e 7 mesi.
Oltre 70 membri di una banda di criminali informatici dediti al phishing, sono stati arrestati in Romania grazie alla collaborazione con le autorità russe.
Considerando che la Russia è sempre stata vista come il paradiso dei cyber criminali, i recenti sviluppi appaiono particolarmente incoraggianti.
"Il crimine online non è più un business esente da rischi", ha aggiunto Hyppönen.
"La cattura da parte delle forze dell'ordine è solo una questione di tempo e speriamo che le notizie di arresti e condanne divengano sempre più comuni".
Per maggiori approfondimenti sugli sviluppi del settore della sicurezza informatica nella prima parte del 2010 è possibile vedere i video: "Mobile Phone Security", "Crime and Punishment" e "Targeted Attacks and Operation Aurora", all'indirizzo www.youtube.com/FSecureNews.

1,3 MILIONI DI ANNUNCI DANNOSI VISTI OGNI GIORNO

La nuova ricerca rilasciato da Dasient (http://blog.dasient.com) indica che in base al campione da loro esaminato, 1,3 milioni di annunci maligni sono visualizzati ogni giorno, il 59 per cento di essi rappresenta drive-by download , seguito dal 41 per cento di falsi software di sicurezza conosciuti anche come scareware.
Inoltre il vettore di attacco, noto come malvertising , è sempre più di tendenza come una tattica scelta per i numerosi attacchi maligni.
La ricerca mostra inoltre altri interessanti risultati:
La probabilità che un utente sia infettato da un malvertisement è due volte maggiore durante un week-end e la durata media di un malvertisement è di 7,3 giorni;
Il 97% dei siti web della lista Fortune 500, una lista annuale compilata e pubblicata dalla rivista Fortune che classifica le 500 maggiori imprese societarie statunitensi misurate sulla base del loro fatturato, sono ad alto rischio di essere infettati con malware a causa di partner esterni (quali i fornitori di widget javascript, le reti pubblicitarie e / o fornitori di software pacchettizato).
Il rischio è così alto, perché il 69% di essi utilizza Javascript esterni per visualizzare porzioni dei loro siti e il 64% di questi siti utilizza applicazioni web obsolete. Le conclusioni della ricerca sono anche supportate da un altro rapporto pubblicato di recente da Google Security Team, secondo cui il falso AV rappresenta il 50 per cento di tutti i malware inviati tramite annunci.

MPAA "fa la guerra" ai soldati americani

La MPAA (Motion Picture Association of America - Organizzazione americana dei produttori cinematografici) ha chiesto, con un documento ufficiale reso pubblico dal Comando Centrale degli Stati Uniti e che può essere visionato all'indirizzo: http://torrentfreak.com/static/pirated-movies-in-iraq.PDF, di prendere provvedimenti nei confronti dei soldati impegnati in Iraq che, per qualsiasi ragione, per vedere programmi e prodotti televisivi e cinematografici, utilizzino DVD contraffatti o file reperiti a mezzo BitTorrent.
Evidentemente la scelta di utilizzare materiale piratato è, in questo caso, dovuta a situazioni contingenti, come la difficoltà di approvvigionamento di materiale originale in un territorio devastato dalla guerra.
Il Comando Centrale degli Stati Uniti ha inoltre sottolineato come non sia possibile negare l'accesso a determinati mercati alla truppe, si parla soprattutto di DVD contraffatti, e neanche proibire tale vendita, dal momento che è regolata dalle leggi locali in materia.
Inoltre, le forze armate affermano che il download illegale è in qualche modo una conseguenza delle carenze dell'industria cinematografica, incapace di rendere accessibili le vie di distribuzione legali all'estero.

Document Approved For Release
By US CENTRAL COMMAND
See FOIA Case #09-0191

UNCLASSIFIED

MNF-I TALKING POINTS:
PIRATED MOVIES IN IRAQ

(PA POSTURE: PASSIVE)

1. PURPOSE:

2. BACKGROUND:

3. THEME:

4. TALKING POINTS

Avanzano i worm USB

McAfee ha presentato il "Report McAfee sulle minacce: primo trimestre 2010", che evidenzia come un worm USB rappresenti il principale malware a livello mondiale. I trend dello spam mostrano che gli argomenti utilizzati dagli spammer nelle loro e-mail variano moltissimo da nazione a nazione, laddove lo spam che promuove diplomi scolastici in arrivo dalla Cina ed altre nazioni asiatiche è in aumento. Le notizie legate ai terremoti e altri rilevanti eventi del 2010 sono stati i principali argomenti per i risultati "drogati" delle ricerche sul web, mentre i server con sede negli Stati Uniti ospitano la maggior parte degli URL malevoli. Le minacce sui dispositivi di storage portatili si sono dimostrate la principale minaccia malware. Le infezioni AutoRun si sono posizionate al primo e al terzo posto data l'ampia diffusione e adozione di dispositivi rimovibili, principalmente drive USB. Una serie di Trojan password-stealing completa la classifica dei primi cinque. Tra questi sono inclusi downloader generici, programmi indesiderati e giochi software che raccolgono dati statistici in modo anonimo. A differenza degli studi passati, la diffusione di queste minacce si è rivelata omogenea in tutto il mondo. Mentre i tassi dello spam rimangono stabili, gli argomenti utilizzati nell'oggetto delle e-mail variano notevolmente da nazione a nazione. Uno dei dati principali emersi da questo trimestre è quello che vede Cina, Corea del Sud e Vietnam ospitare la maggior quantità di spam legato ai diplomi scolastici, che promuovono l'acquisto di documenti contraffatti per comprovare l'idoneità a ricoprire determinati impieghi. Singapore, Hong Kong e il Giappone vantano tassi eccezionali di spam legato ai messaggi di notifica dello stato di consegna dei messaggi indicando un possibile problema legato alle funzionalità di filtering preventivo della posta. McAfee ha inoltre rilevato che Tailandia, Romania, Filippine, India, Indonesia, Colombia, Cile e Brasile vantano una quantità più elevata di infezioni malware e di spam. Queste nazioni hanno sperimentato una significativa crescita di Internet negli ultimi cinque anni e mancano di consapevolezza relativa alla sicurezza. Gli aggressori sfruttano i principali eventi di cronaca per "drogare" le ricerche effettuate su Internet. I terremoti di Haiti e del Cile sono in cima all'elenco (N. 1 e N. 2, rispettivamente). Il ritiro delle auto da parte di Toyota, l'Apple iPad e la settimana delle finali NCAA del campionato universitario di basket statunitense seguono a ruota. Sfruttando la tecnica denominata manipolazione dei motori di ricerca, i criminali informatici continuano ad utilizzare analitiche e logica di classificazione delle pagine per sfruttare i termini di ricerca più frequenti e guidare il traffico su siti web malevoli. Con ben il 98%, gli Stati Uniti risultano ospitare la maggior parte dei nuovi URL malevoli nel primo trimestre del 2010. L'enorme quantità di nuovi URL malevoli con sede negli Stati Uniti è dovuta all'ubicazione di molti servizi Web 2.0 diversi, la maggior parte dei quali vengono forniti da ubicazioni negli Stati Uniti. Per quanto riguarda il rimanente 2%, la Cina ne ha ospitati il 61% mentre il Canada ne ha ospitati il 34%.

L'HACKER ENTRA IN AUTO

Secondo uno studio realizzato da CAESS (Center for Automotive Embedded Systems Security) e presentata ad Oakland durante il simposio IEEE sulla sicurezza e la privacy a cui hanno partecipato i ricercatori delle Università statunitensi di Washington e San Diego, l'elettronica sempre più sofisticata delle auto ne metterebbe a rischio la sicurezza. Secondo questo studio molte automobili possono essere manomesse connettendo un computer alla porta per la diagnosi elettronica (OBD) e agendo in remoto da un altro PC. Le modifiche indotte con l'intervento del Pc possono avere conseguenze disastrose: impossibilità di usare i freni, disabilitati tramite la centralina dell'ABS, spegnimento del motore mentre si sta guidando, blocco delle porte, errate indicazioni sulla velocità e malfunzionamento dei dispositivi relativi alla sicurezza dell'auto. L'attacco, inoltre, può arrivare per via telematica anche attraverso la CAN, la rete che collega tutti i dispositivi elettronici dell'auto. Lo studio può essere scaricato all'indirizzo: http://www.autosec.org/pubs/cars-oakland2010.pdf

ADOBE SOTTO TIRO

Secondo il rapporto pubblicato da Kaspersky Lab 'Information Security Threats in the First Quarter of 2010', i prodotti Adobe sono diventati il primo obiettivo per hacker e virus writer in tutto il mondo grazie alla loro vasta diffusione e alle loro caratteristiche multi-piattaforma. Gli utenti dei prodotti Adobe sono spesso ignari delle minacce potenziali in cui potrebbero incorrere aprendo un file PDF di origine sconosciuta. Tra le molte varietà di exploit che sono state rilevate, la famiglia Exploit.Win32. Pdfka è risultata essere quella più diffusa con una percentuale del 42,97%. Questo exploit sfrutta le vulnerabilità di Adobe Reader e Adobe Acrobat.
Sommando Exploit.Win32. Pdfka ed Exploit.Win32.Pidief, due famiglie di exploit che colpiscono frequentemente i prodotti Adobe, si raggiunge una percentuale totale del 47,5% che rappresenta quasi la metà di tutti gli exploit rilevati. Questi exploit sono documenti PDF che contengono codici Javascript che, ad insaputa dell'utente, scaricano e lanciano altre applicazioni malware direttamente da Internet. Il rapporto evidenzia inoltre che molti utenti di prodotti Adobe non hanno ancora installato le patch progettate per rimuovere le vulnerabilità del software rimanendo così esposti a potenziali attacchi. Tra le prime dieci vulnerabilità dei software più diffusi rilevate sui computer degli utenti nei primi tre mesi del 2010, tre sono stati trovati in prodotti Adobe, sei sono stati trovati nei prodotti Microsoft, ed uno è stato trovato in un prodotto Sun. Le tre vulnerabilità che colpiscono i programmi Adobe sono state trovate nel 23,37%, 17,87% e 15,27% dei computer esaminati, si tratta di vulnerabilità che permettono agli hacker di avere il pieno controllo su un sistema in modalità remota. Per rimuovere una delle vulnerabilità nei prodotti Adobe, diventata di dominio pubblico da oltre tre anni, è stata creata una patch che è rimasta disponibile per lungo tempo, e ciò a dimostrazione che molti utenti continuano a non aggiornare il loro software. Per risolvere questo problema, il 13 aprile Adobe ha lanciato un servizio automatico di aggiornamento che viene eseguito in background. Gli sviluppatori sperano che ciò contribuirà a ridurre il numero di applicazioni sprovviste di patch, le più ambite dai i criminali informatici.

Riccardo Meggiato
redazione@hackerjournal.it

MESSAGGI AL SICURO SU IPHONE, QUASI PER GIOCO

Privacy dei nostri dati? Oh, ci sono molti sistemi per proteggerla, anche se la totale sicurezza non è cosa di questo mondo (e noi lo sappiamo bene...). Senza contare che si deve sempre trovare un buon compromesso con la praticità. Prendiamo un comune messaggio di testo, per esempio: esistono algoritmi e software in grado di crittografarlo in modo molto efficiente, ma di rado hanno la velocità e l'intuitività necessaria per farlo in tutte le situazioni. Il telefono ne è un facile esempio: che tragedia crittografare i messaggini scritti in fretta e furia!
Tuttavia, se utilizziamo un iPhone, una discreta soluzione viene da Secret Crypto Wonder Badge, orribile nome di una "app" altrimenti apprezzabile. Con la scusa di un'interfaccia giocosa, questo software replica in buona sostanza il "disco cifrante" di Leon Battista Alberti. Un momento: di cosa si tratta? Conosciuto come il primo sistema di cifratura polialfabetica, il disco cifrante su descritto nel "De Cifris" attorno al 1467.
Questo dispositivo è composto da due dischi, uno che riporta le lettere dell'alfabeto, e numeri, nell'ordine corretto, e l'altro con i caratteri mescolati alla rinfusa. In questo modo è facile far corrispondere una lettera "sbagliata" a quella corretta di una parola. Ora, è ben vero che non si tratta di un sistema crittografico con chissà quale livello di sicurezza, e non possiamo certo portarci appresso un bel disco cifrante, ma è altrettanto vero che un software facile, pronto a offrire una cifratura "mordi e fuggi", fa sempre comodo.

CRIPTO

Una app che offre un sistema crittografico old-style, ma efficiente e pure divertente

SICUREZZA A POCHI CENTESIMI

Secret Crypto Wonder Badge si scarica e installa dall'app store, elargendo un modesto obolo di 0,79 euro. Un costo risibile, che forse non dà giustizia alle potenzialità dell'applicazione. Questa, una volta avviata, ci fa scegliere quale dei sistemi di cifratura utilizzare, tra Alpha,

Il Disco Cifrante rivive in chiave digitale, grazie ad una app per iPod Touch e iPhone che lo sfrutta a dovere.

Una volta impostata la corrispondenza tra i due dischi, si crea un vero e proprio sistema crittografico personalizzato.

Nessuno spazio, in nome della massima sicurezza! Il trattino, infatti, garantisce una stringa continua di numeri, più difficile da decifrare.

Quale oscuro messaggio è nascosto in questa sequenza di numeri? Scoprirlo è molto facile e veloce, se si ha la giusta chiave crittografica!

Bravo e Charlie. Fatto questo, eccoci davanti al mitico cifrario, in versione iPhone.
Giriamo i dischi selezionando la corrispondenza desiderata tra lettere e numeri e, per finire, tocchiamo il lucchetto centrale. A questo punto tocchiamo, in basso, su Encode. Si apre una finestra di testo dove scrivere il messaggio da cifrare. Per rendere ancora più complessa un'eventuale decifrazione da parte di estranei, e come del resto avviene con alcuni dei più noti sistemi crittografici, le parole non possono essere separate dal classico "spazio". Al suo posto, comunque, c'è un più che valido "trattino". Una volta inserite le parole desiderate, tocchiamo in basso, su Engage, ed ecco ottenuta l'agognata stringa crittografata.

Una serie di numeri che, pur non mettendoci al riparo dagli critto-analisti più scafati, offre una discreta protezione contro occhi troppo curiosi. Senza contare che, anche ai più esperti, a volte non viene in mente che una sequenza sconclusionata di numeri può nascondere, in realtà, un messaggio…

CRITTOGRAFIA VIA EMAIL

Una volta effettuata la codifica del testo, la utilizziamo direttamente dove necessario o, volendo, la inviamo al nostro contatto via e-mail. Per farlo, basta toccare direttamente l'icona a forma di e-mail e procedere come indicato.
Il destinatario ottiene così il messaggio cifrato e, per effettuare la decodifica, gli è sufficiente utilizzare una copia di Secret Crypto Wonder Badge, sfruttando i dati che accompagnano la missiva elettronica. Decodificare un messaggio con questo software, del resto, è un gioco da ragazzi: dopo aver toccato il simbolo del lucchetto, è sufficiente toccare la voce Decode e inserire la sequenza di numeri, toccando infine su Engage. Siamo ben consapevoli che il livello di sicurezza offerto da questo programma per iPhone non è certo comparabile con sistemi crittografici avanzati, tuttavia esistono informazioni per le quali è più che sufficiente. Insomma, non avremo sempre a che fare con progetti bellici top-secret o guerre cibernetiche no? Scherzi a parte, calcoli alla mano, Secret Crypto Wonder Badge, con i suoi tre set di corrispondenze, consente la bellezza di 2700 diverse combinazioni: davvero niente male.
Nel caso, per una app dall'interfaccia più seriosa, e con una tecnologia un filo più efficiente, ci possiamo rivolgere a CryptoMail. Sviluppato dalla nota ReynoldTech, è un software che, come il nome fa intuire, è dedicato alla posta elettronica, anche se la macchinosità dei comandi ne limita un po' l'immediatezza. Del resto è disponibile gratuitamente, quindi provarlo non costa davvero nulla.

EVENTI

di penelope.di.pixel
redazione@hackerjournal.it

HACKIT 0X0D FIGHT CONTROL

HACKMEETING

L'APPUNTAMENTO PER L'ORMAI DECENNALE HACKMEETING È FISSATO QUEST'ANNO PER IL 2 LUGLIO A ROMA

L'incontro annuale della lista Hackmeeting si svolgerà quest'anno a Roma il 2-3-4 luglio. Cercheremo di seguire lo sviluppo di questo evento, che nei suoi dieci anni di vita intorno alla diffusione/condivisione orizzontale e libera di conoscenza ha costruito pratiche sociali, culturali e tecnologiche portatrici di un agire politico radicalmente nuovo.

IL TEMA

Fight Control: comprendere le forme del controllo contemporaneo per trovare e inventare strumenti e spazi di resistenza. Dalle problematiche squisitamente tecnologiche come protocolli, p2p, privacy, sicurezza, controllo per HACKIT 0X0D significa anche interrogarsi su temi come censura, lavoro, consumo, social network per arrivare a parlare di cibo, acqua e filiere produttive. Il tema è caldo e attuale come sempre.

I LUOGHI

Scelto per la sua storia e le caratteristiche fisiche, sarà il CSA La Torre a ospitare la tre giorni capitolina del meeting: nel quartiere di Casal De' Pazzi tra Ponte Mammolo, Talenti e Montesacro, una vecchia villa immersa nel verde dove sarà possibile allestire tende e campeggi... e connettersi.
La rete, a quanto pare, è già pronta.
Anche se per Hackmeeting non sarebbe un problema: in altre occasioni, mettendo insieme cavi e competenze, le infrastrutture sono state create da zero.
Incontri e riunioni di preparazione si sono invece svolti fra il CSO Strike, l'ass. Fusolab e il Forte Prenestino, intrecciando geografie metropolitane, storie e percorsi di attivismo politico e sociale ma soprattutto persone che insieme discutono e lavorano per dare corpo e vita all'evento.

WARM UP

Come per l'edizione 2009, anche quest'anno riconfermata la formula warm up: una serie di incontri di approfondimento che costruiscono le tappe di avvicinamento al meeting, nell'idea di aprirsi e coinvolgere il territorio. Fra gli

attivatori dei warm up, il BugsLab che fra fine aprile e maggio organizza a Strike dibattiti sui DRM, tutela dei dati in rete servizi autogestiti, P2P, utilizzo consapevole del file sharing e social network.
Le facoltà di Chimica, Fisica e Matematica della Sapienza, che ospitano incontri su free software e pubblicazioni in ambito scientifico, filosofia e comunita' nel software libero, riuso e trashware. La rete Ninux, con il suo seminario sulle reti mesh al Fusolab.
E Officine Naturali che al Forte Prenestino si cimenta in un laboratorio di erboristeria DYI con tanto di escursione nel verde a caccia di piante officinali. Per tenersi aggiornati sul programma: http://it.hackmeeting.org

IL PRE HACKMEETING

Sabato 15 maggio si svolge infine a Firenze il pre-hackmeetig, legandosi a Do It Your Trash: una tre giorni dedicata alla cultura del riciclaggio con workshop, incontri e presentazioni, il cui sottotitolo recita "We will survive?".
Una scelta politica precisa dove la tecnologia incontra lotte ambientali, guerriglia gardening, borse a km 0, arte riciclata contro il nucleare e non avrebbero potuto scegliere meglio.

Hackmeeting si svolgerà quest'anno a Roma il 2-3-4 luglio. Si tratta di un evento ormai decennale che guida il visitatore all'interno dei temi della diffusione/condivisione orizzontale e libera conoscenza.

:: POSTA ::

VITA AI VECCHI PC

La nostra economia è seriamente in difficoltà e due nuove risorse gratuite, dal mondo informatico, consentono di risparmiare grazie ad inventiva e pazienza. Si tratta di due diverse distribuzioni Linux, una per il recupero di vecchi pc e l'altra pensata per supportare le aziende ed il lavoro senza pagare una qualsiasi licenza, ecco la pagina web di riferimento: http://ogigia.altervista.org/index.php?mod=none_Linux_Ogigia_sommario
dove trovate tutte le informazioni, e che restera' a questo indirizzo Internet per gli anni a venire per aiutarvi a linkare questo grande regalo.
Anche se non usate questi due sistemi Linux (italianizzati e semplificati) non importa, facendo sì che altri ne vengano a conoscenza permettete di aumentare le risorse e quindi la ricchezza del nostro paese, contribuendo a far girare i soldi in Italia.
Nel 2003 dall'Italia sono andati alla Microsoft più soldi del decifit dello stato, non si capisce perché, considerando anche lo stato di crisi crescente, dobbiamo continuamente mandare i nostri euro fuori dalla nostra nazione ad arricchire un tizio che è già più ricco dell'intera Africa...
Aiutateci ad aiutare tutti e a risparmiare denaro, fate sapere a tutti che i pacchetti Linux Ogigia sono stati pensati proprio per lenire un poco la carenza economica generale.
Il webmaster del portale Ogigia.

Diamo sicuramente spazio alla tua iniziativa perché ci sembra lodevole. Noi siamo assolutamente convinti che il mercato debba orientarsi verso soluzioni open source, il problema è che dieci anni fa si pensava che il mondo, quello informatico, non fosse ancora maturo per soluzioni open. Oggi, osservando il successo di tutta una serie di soluzioni chiuse, vedi iPhone e iPad, si sta insinuando il dubbio che forse non lo sarà mai...

QUESTIONE ARRETRATI, NON TROVATI E VARIE

Avevo lasciato detto al mio edicolante "di fiducia" di lasciarmi sempre i vostri numeri. Per qualche strana ragione ho perso i numeri 197 - 198 - 199. Suppongo che non mi lascerà nemmeno il numero 200. Vi chiedo cortesemente di potermeli inviare. Potrei pagarli con contrassegno. Accetto anche se mi pagate ogni rivista al costo di 3,00 euro. Vi prego, ci tengo tantissimo. Non ho mai perso nemmeno un vostro numero da quando ho conosciuto la vostra rivista, e ci terrei ad averla in copia cartacea, così come ho le altre.
Vi ringrazio per il tempo concessomi.
Mi auguro che soddisferete la mia richiesta.
Distinti Saluti.

Giuseppe

Torniamo ancora per una volta sull'argomento arretrati e dintorni. Purtroppo, considerala se vuoi una questione di coerenza ma forse è solo un'impostazione iniziale che non è mai stata corretta, non è possibile richiedere arretrati, né abbonarsi. Stiamo studiamo una formula di abbonamento on-line, per leggere/ricevere la rivista sul computer.
Al momento l'unico servizio che offriamo è quello della lettura dei pdf della rivista che sono scaricabili del tutto gratuitamente dal nostro sito www.hackerjournal.it.
La pubblicazione sul sito evidentemente non è contestuale a quella dell'edicola, il materiale viene messo a disposizione dopo un ragionevole lasso di tempo per non creare una concorrenza diretta col numero cartaceo. Però tutti i numeri vengono caricati e questo può considerarsi, a pieno titolo, l'archivio digitale di HJ. Capisco che il cartaceo abbia un altro significato e, soprattutto, un'altra consistenza. C'è anche il discorso della collezionabilità e, forse, di una maggiore facilità di consultazione. Però al momento meglio di così non ci riesce di fare. L'altro consiglio è di trovare un'edicola di riferimento a cui chiedere di farsi recapitare dal distributore una copia della rivista e acquistarla sempre lì in modo che si possa creare una sorta di fidelizzazione e l'edicolante, sia avendo la certezza di venderla, sia stimolato a richiedere una o più copie di HJ.

DECODIFICARE LE PASSWORD DI WINDOWS

Salve vorrei segnalare il mio sito www.decryptyourpass.com che permette di decodifcare le password di windows, il sito può essere collegato ad un articolo della rivista che spiega l'algoritmo lm hash e come grazie al sito segnalato è possibile decodificare le password di windows.
Grazie.

Intanto diamo notizia del sito, che ci sembra interessante. Esistono diverse soluzioni open source pwdump che mostrano gli hash delle password LM e NTLM dell'utente locale dal database SAM (Security Account Manager). Faremo una prova su strada della tua soluzione e magari invitiamo i lettori a fare altrettanto.

CARTA CANTA

Cara Redazione di HJ, sono molto contento di questo restyling che ha avuto la rivista, ad esempio si trattano gli argomenti meglio di prima, c'è abbastanza spazio per un argomento che ne ha necessità ma c'è solo un problema. Da quando la rivista è stata ridisegnata avete cambiato il tipo di carta ... sembra "carta plastificata" perchè non tornate ad utilizzare quella riciclata stile cartoncino... so che può essere una stupidaggine è posso sembrare un co**1*0*3 ma volevo sapere il perché di questa scelta.
Saluti Esultanti.

:: POSTA ::

La carta è legata a scelte dell'editore. Ci possono essere varie ragioni: disponibilità di un certo tipo carta in quel momento, convenienza e altro. Teniamo comunque in buona considerazione la tua segnalazione.
p.s. Ricambiamo i saluti Esultanti.

CORSO DI PROGRAMMAZIONE IN C

Mi chiamo Marco, ho dodici anni e abito a Milano. No, non sono il tipico lamer che vuole dominare il mondo e se oggi vi scrivo non è per chiedervi come rubare una password di MSN, ma solo per dare consigli riguardo il corso di C. Purtroppo non ne farò un grande utilizzo conoscendo già questo linguaggio, ma non per questo è da scartare (un ripasso ci sta sempre), anzi la trovo una bella iniziativa, brillante e ben sviluppata. Sul forum, che visito regolarmente anche se non iscritto, ho letto anche come è suddivisa. Proprio riguardo a queste mi sono venute alcune idee, non brillanti, ma pur sempre idee. Ve ne elenco alcune, okay?

Come prima cosa proporrei di parlare prima delle stringhe, e dei vari metodi per manipolarle, e solo poi degli Stack e dei nuovi tipi di dato ottenuti attraverso strutture e puntatori.

Poi, sempre parlando dei nuovi tipi di dato, io parlerei, oltre agli Stack, anche degli alberi binari. E' una struttura ricorsiva e le funzioni per manipolarle sono ricorsive (un buon momento per spiegare cosa significa nei dettagli). Diciamo anche che ha una struttura precisa e ordinata che impedisce la creazione di duplicati e che facilita la ricerca di elementi all'interno della stessa. Insomma, mi sembra un argomento da non perdere assolutamente!

Per ultimo proporrei di aggiungere un nuovo capitolo, alla fine, prima degli esercizi. Un capitolo riguardo all'OOP... in C. Assurdo? No. Ma in C non esistono le classi e non è possibile programmare ad oggetti! Vero, ma anche no. Il C è un linguaggio procedurale, d'accordo.

Questa non ci vieta, però, di usare, sudando ovviamente, l'OOP. Ok, ma come si fa? Ottima domanda. Con questo vi rimando a Google e vi consiglio la lettura di http://www.python-it.org/forum/index.php?topic=4033.0. L'ultima domanda potrebbe essere Perché usare l'OOP in un linguaggio come il C, dove non è supportata. Bene, l'OOP è la tecnica di programmazione più usata al momento. Inoltre Linux è fatto in C, con largo utilizzo di questa tecnica. Ma la vera utilità di imparare l'OOP in C sta nel fatto che, in questo modo, si riescono a capire bene tutti i concetti nascosti dietro questo metodo, e, di conseguenza, imparare nuovi linguaggi orientati agli oggetti (Python, C#, Java, ecc...) risulterà banale. Imparare questa tecnica in C aiuta poi nella programmazione in generale. Si inizia a vedere un programma come una cosa astratta.

Infine, se volete vedere un'altra implementazione dell'OOP potete guardarvi i sorgenti di Python (include/Object.h). E poi, Google è vostro amico...

Marco

Il tuo rilievo ci sembra davvero interessante. Giriamo senz'altro le tue riflessioni agli autori che stanno alacremente lavorando alla realizzazione delle varie puntate su cui si articola il Corso di programmazione in C. Peraltro lo spunto su OOP ci sembra comunque interessante anche considerandolo fine a se stesso. Molti lettori, dalla lettura della tua mail, potrebbero essere stimolati ad approfondire l'argomento confidando proprio sull'"amico" Google.

QUALCHE APPUNTO

Piacere, io sono un lettore del vostro giornale, purtroppo da poco mi sono trasferito e sono riuscito solo ultimamente a convincere la mia edicolante a prendermi una copia del vostro giornale e quindi mi sono perso i primi 2 numeri con le lezioni di programmazione in C.
Visto che era da un po' che volevo imparare questo linguaggio, mi è dispiaciuto. Quindi volevo chiedere se mi potevate i PDF dei due numeri con le prime lezioni.
Naturalmente ve le pagherei, magari anche in anticipo tramite poste pay.
Infine vorrei complimentarmi per la rivista molto ben strutturata secondo me^^.
L'unico difetto secondo me, ma che ho riscontrato solo alcune volte, anche se per la maggior parte di queste era forse necessario, è che vengono date per scontate alcune cose, come nel numero 197 mi sembra, nell' SQL injection l'atricolo parte dicendo "questa è la nostra pagina vulnerabile", il problema è che io non so come raggiungere la pagina vulnerabile... Comunque ho riscontrato questa cosa anche nelle varie guide sulla rete che ho trovato. naturalmente questa è solo la mia opinione, ma prendetela come una critica costruttiva.

Purtroppo per la rivista vale quanto già detto in precedenza: non inviamo arretrati e non facciamo abbonamenti. Puoi ricorrere al sito www.hackerjournal.it per cercare di recuperare del materiale. Per quanto riguarda invece il fatto che alcune cose siano date per scontate, è probabilmente riferibile più che altro a un problema di spazio. Non avendo molte pagine a disposizione cerchiamo di arrivare subito al "nocciolo" del problema, sacrificando un po' di premessa. E' proprio un problema con cui si stanno confrontando i nostri collaboratori del Corso in C.

di Massimiliano Brasile
redazione@hackerjournal.it

BUCHIAMO FACEBOOK

ATTACKING

Ti fidi di Facebook? Ecco come bucare il social network

Diciamocelo: nolenti o volenti i social network sono un successo planetario che prescinde dalla qualità e quantità di cose scritte fuori e dentro. Facebook (FB) è forse il fenomeno più clamoroso, per la crescita continua di persone connesse da tutto il mondo e i fenomeni di aggregazione che raccolgono individui fisicamente e culturalmente distanti.
Ma FB è pur sempre un'applicazione software e come tale soggetta a bug, falle di sicurezza e anomalie. Vediamo cosa si rischia quindi a utilizzarlo e a comunicare al social network foto e informazioni personali.

HACK DI UN INVITO

Può capitare che se usiamo da tempo internet abbiamo diversi indirizzi e-mail che nel tempo abbiamo dato ad amici, colleghi, parenti e così via e che quindi non tutti si siano sincronizzati con l'ultimo account che stiamo usando come ufficiale, ad esempio su Gmail. Disgrazia vuole che qualcuno con cui desideriamo assolutamente diventare amici su FB ci invii l'invito proprio su un account che vorremmo chiudere, ma non possiamo. Potremmo provare ad aggiungere il nostro amico al nostro profilo FB cercandone nome, cognome e-mail. Ma potremmo anche ottenere centinaia di risultati, o nessuno perché magari ha scelto uno pseudonimo e non i dati reali. Perdiamo quindi la possibilità di agganciarlo al nostro network dove siamo registrati con un indirizzo diverso? Per nulla, basta un minimo di hacking.
L'URL che ci ha inviato FB via e-mail ha un formato simile a questo:
http://www.facebook.com/p.php?i=XXXXXXXXX&k=YYYYYYYYYY&r&v=2
Dove XXXXXXXXX rappresenta l'ID su FB dell'utente che ci ha contattato. Con questa informazione possiamo raggiungere direttamente il suo profilo su FB sostituendolo all'indirizzo sotto http://www.facebook.com/profile.php?id=XXXXXXXXX
Una volta aperto, clicchiamo su "Add as Friend" e dopo essere stati accettati dal nostro amico possiamo cestinare il vecchio invito. La protezione è effettivamente debole, trasmettendo all'esterno del social network un identificativo privato. Restano tuttavia le difese interne basate sull'autenticazione dei due utenti coinvolti.

CRACK DELLA PASSWORD

Qualcuno si aspetterà di venire a sapere prima o poi che ci sia una grave falla di sicurezza o magari grazie a una mano data

da Microsoft, siano state lasciate delle vulnerabilità intrinseche che permettono di accedere liberamente agli account di FB. Invece senza troppe speculazioni, è stato realizzato un piccolo tool scritto in Java che fa il suo dovere alla vecchia maniera, con un brute-force.
Può utilizzare vocabolari di parole già pronte ed è gratuito. Si chiama Faceboozz (http://facebooz.goldeye.info) e gli occorre solamente un Java Runtime-Environment per funzionare localmente. Si inserisce l'e-mail della vittima, si carica il dizionario che si vuole utilizzare e lui parte, incurante delle pause generate dalla controffensiva di FB che allungheranno i tempi, ma si sa che la pazienza è la virtù dei forti! Questo per ricordare che una password non è mai troppo complicata! Ed è bene usare anche i caratteri più strani creando parole o frasi senza senso.

CONTROLLO DI UN ACCOUNT

Esistono poi metodi più raffinati per sconvolgere la tranquillità di un utente FB, che basano l'attacco sui cookie che sono generati man mano che utilizziamo il social network per prendere contatto con gli altri utenti. In pratica può essere realizzato un vero sniffing di credenziali d'accesso da remoto basandoci sulle informazioni che FB ci permette di ottenere sulla vittima. Il risultato finale può essere quello di prendere il controllo di un account su FB senza conoscere la password di accesso e senza che la vittima se ne accorga.
Per avere i cookie ogni metodo può andare bene: sniffando il traffico, XSS, social engineering, ARP Poison-Sniffing. Basta loggarsi in FB con il proprio account e sniffare i propri cookie o raccoglierne di nuovi man mano che entriamo in contatto con l'obiettivo.

Alla fine avremo raccolto i dati necessari da dare in pasto a un tool abbastanza potente chiamato FBController (http://my.opera.com/quakerdoomer/blog/fbcontroller-v3-0-facebook-

FBController somiglia molto ai tool dei black-hat, ma l'autore precisa che non viene effettuato alcun crack dell'account.

Passando un file che contiene i cookie FBController riesce a sferrare i suoi attacchi mirati.

In questo esempio viene mostrato come è possibile avviare una chat al posto della vittima.

Semplice ed efficace, Faceboozz utilizza il meccanismo di errore di FB per tentare più volte di accedere all'account; l'utilizzo dei dizionari permette di tarare l'attacco sulla vittima.

In questo esempio viene mostrato come sia possibile alterare i settaggi scelti dalla vittima in merito alla protezione della privacy

Nexus è un plugin molto efficiente che riesce a ricostruire la rete di connessioni che potrebbero legarci a un utente veramente distante da noi.

control-utility-version-3-0). Tramite questo strumento è possibile prendere il controllo di qualunque account ad esempio per controllarne la lista degli amici online o avviare una conversazione come fossimo quell'utente e soprattutto inviare e ricevere messaggi al suo posto.

E' possibile vedere la cronologia delle conversazioni passate dell'utente vittima con I suoi amici a prescindere che essi siano online o offline, effettuare "Poke" ai suoi amici e modificare i parametri relativi alla Privacy.
Nella versione 3.0 di FBController viene poi inserito il Facebook Nexus Explorer (attualmente in beta), che funzionando come un motore di ricerca, permette di individuare utenti di FB a partire dalle liste di amici, restituendo il numero di salti necessari a raggiungere gli utenti trovati. In FB esiste una funzionalità simile chiamata "Mutual friend" che viene però limitata a un solo salto, mentre con Nexus non ci sono limiti (nella beta attuale il limite è temporaneamente a uno comunque).

Il funzionamento è molto semplice: basta definire l'ID di partenza e un ID obiettivo, oltre a indicare la profondità della ricerca che si intende raggiungere. Per ogni nuovo ID raggiunto vengono estratte le liste di amici e possono partire nuove ricerche in modo ricorsivo. Questo quindi negando di fatto il controllo di FB sulle relazioni, la chiave centrale del funzionamento dei social network.

CONCLUSIONI

Questa piccola panoramica voleva soltanto mostrare alcune delle debolezze inevitabili di un social network come FB.

E ricordando che nel campo degli OS, il progressivo affermarsi di Windows come sistema desktop è coinciso con l'aumento esponenziale di virus e attacchi diretti verso questa piattaforma, assisteremo molto presto al propagarsi di minacce serie che funzionano grazie alle API dei social network e riescono a infettare molte più vittime, spesso poco esperte e quindi più vulnerabili, in brevissimo tempo.

Agli amministratori di rete l'arduo compito di far conciliare informazione e prevenzione degli utenti per i quali si è responsabili. Alle riviste come la nostra quella di suonare campanelli di allarme.

Guido Camerlingo AKA Guiz
www.guiz.it/

DISTANZA DI LEVENSHTEIN

Implementiamo il forse cercavi di Google

INTERNET

L'ALGORITMO DI RICERCA DELLA PAROLA IPOTETICAMENTE CORRETTA SI BASA SULLA DISTANZA DI LEVENSHTEIN, VEDIAMO COME FUNZIONA.

Il "forse cercavi" di Google è una funzionalità della ricerca molto comoda nel caso di "errori" di battitura o semplice dubbio. Vediamo come riprodurla per le ricerche dei nostri siti. L'algoritmo di ricerca della parola ipoteticamente corretta si basa sulla distanza di Levenshtein anche nota come 'Edit Distance'.

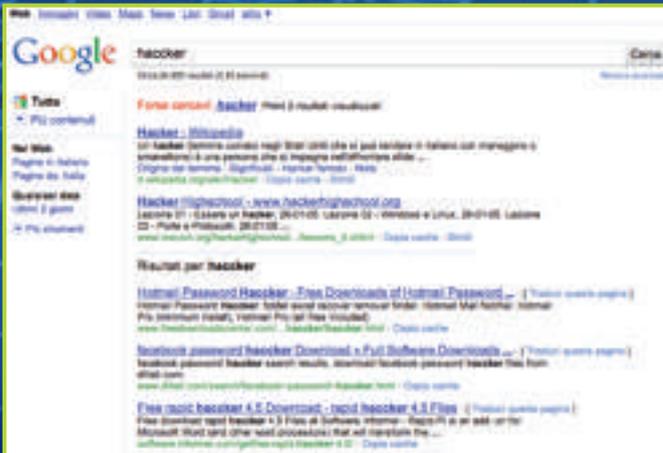
LA DISTANZA DI LEVENSHTEIN

La distanza di Levenshtein è il numero di modifiche elementari necessarie per trasformare una stringa A in B.
Le modifiche sono:
Sostituzione di un carattere
Cancellazione di un carattere
Aggiunta di un carattere
Mettiamo caso che nella nostra ricerca scrivessimo erroneamente linxu.
Per trasformare "linxu" in "linux" sono necessarie n modifiche. Questa n è la distanza di Levenshtein.

La distanza quindi sarà
Levenshtein('linxu','linux')
Minore è il numero che esprime questa distanza maggiore sarà l'attinenza tra le parole.
Per far funzionare il tutto abbiamo bisogno di un enorme dizionario italiano: ne ho estrapolato uno abbastanza completo eseguendo un 'merge' di diversi dizionari.
Il problema è che dovremmo eseguire il test per ogni parola presente nel dizionario. Il che diventerebbe molto pesante poiché, avendo circa 145000 parole, ci sarebbero 145000 distanze da calcolare.
La soluzione consiste nell'effettuare un primo filtro nella selezione delle parole; quindi utilizzare un database (nel nostro caso MySQL) per estrarre solo le parole che hanno una lunghezza pari alla parola cercata ± un valore numerico a nostro piacimento, per esempio 1.
Quindi estrapoleremo dal database
- nel caso di linxu = 5 char (caratteri)
- tutte le parole che hanno lunghezza di 4,5,6 e le confronteremo con la distanza di Levenshtein.
Per velocizzare ulteriormente l'estrazione dal dizionario indicizziamo i campi "parola" e "lunghezza parola".

IL NOSTRO DIZIONARIO

Il nostro dizionario è composto da 143770 parole. I campi del database sono organizzati così:
N_REC_ID - Progressivo del record
C_PAROLA - Parola
N_CHRLEN – Intero che rappresenta la lunghezza della parola
E' disponibile un dump del database con le 143770 parole al seguente indirizzo (Oltre al SQL è in CSV,TXT e XLS):

http://www.guido8975.it/index.php?ctg=6&id=67

ATTO PRATICO

Questo script si basa su 3 file:
conf.php
Solito file di configurazione e connessione al database
ricerca.php
File della ricerca
proc.php
File dell'algoritmo

FILE CONF.PHP

```
<?php
    $server="localhost";
    $user_n="root";
    $password="password";
    $db_name="dizionario";
    $connessione = mysql_
connect($server, $user_n,
$password)
    or die("Connessione non
riuscita: " . mysql_er-
ror());
    mysql_select_db($db_
name, $connessione)
    or die ("Errore nella
selezione del database.");
?>
```

FILE RICERCA.PHP

La parola è inserita,in questa pagina a scopo illustrativo in una variabile $rRic a cui possiamo assegnare invece di 'linxu' i dati provenienti da un POST con $_POST['nomecampopost'].

```
<?php
    include 'proc.php';
    $rRic = trim('linxu');
    if(!empty($rRic)) {
      $rRicRes = "";
        $rRicArr = explode("
", $rRic);
        foreach ($rRicArr as
$j) {
          $rRicRes .=
LevWord($j)." ";
        }
        $rRicRes = substr_
replace($rRicRes ,"",-1);
      }
      if($rRicRes <> $rRic){
echo "Forse cercavi: ".$rRi-
cRes;}
      //Ricerca vostro sito
      //Qui aggiungerete la
parte di codice per effet-
tuare la ricerca nel vostro
sito.
?>
```

FILE PROC.PHP

Qui è presente la funzione che calcola la distanza di Levenshtein. Modificando la variabile $rSrcDis, che di default è a 1, possiamo aumentare lo spettro di ricerca della parola nel db. Maggiore è il numero maggiore sarà lo spettro di ricerca.

```
<?php
function
LevWord($SearchPostWord){
    include 'conf.php';
    $rSrcDis = '1';
    $SearchPostWordLen =
strlen($SearchPostWord);
    $query_rch = "SELECT *
FROM italiano WHERE N_CHRLEN
in (".($SearchPostWordLen
$rSrcDis).",".$SearchPostWor
dLen.",".($SearchPostWordLen
+$rSrcDis).") ";
    $result_rch = mysql_
query($query_rch, $conne-
sione);
    $rWordLevDst = -1;
    while($row_rch = mysql_
fetch_array($result_rch)){
        $rLevDst = leve
nshtein(strtolower($SearchP
ostWord), strtolower($row_
rch['C_PAROLA']));
        if($rLevDst ==
0) {
            $rWord =
$row_rch['C_PAROLA'];
            $rWordLevDst
= 0;
            break;
        }
elseif($rWordLevDst < 0 ||
$rLevDst <= $rWordLevDst) {
            $rWord =
$row_rch['C_PAROLA'];
            $rWordLevDst
= $rLevDst;
        }
    }
    if(!empty($rLevDst)){
return $rWord;}else{ return
$SearchPostWord;}
    mysql_close();
}
?>
```

CONCLUSIONE

L'esempio completo è disponibile sul sito http://www.guido8975.it/index.php?ctg=6&id=67 .Dove sono disponibili sorgenti e dump del database. I sorgenti scaricabili sono anche commentati. Un esempio del funzionamento è implementato in http://www.geek-blog.it/ nella ricerca.

CYBERENIGMA

HACKER JOURNAL 204

Diciamoci la verità, la crittografia ci ha un po' stufati. Sarebbe il caso di divertirci con qualcosa di nuovo.
Durante le cinque parti finora pubblicate del corso di programmazione in C abbiamo dato alcune definizioni relative ai rischi connessi all'utilizzo di determinate metodologie di sviluppo dei nostro software in C.

Per ogni argomento analizzato abbiamo poi corredato la spiegazione con alcuni spezzoni di codice esemplificativo. Obiettivo del Cyber Enigma di questo numero è pertanto lavorare con i sorgenti offerti in tutte le parti del Corso finora pubblicate.

LA SFIDA:
Newbie: Individuare il codice e per ogni blocco vulnerabile individuato spiegare la problematica.
Mid: Correggere il codice vulnerabile facendo uso delle nozioni apprese durante la lettura del Corso.
Esperti: Scrivere almeno un exploit capace di sfruttare eventuali vulnerabilità presenti nel codice presentato in tutte le parti finora pubblicate del Corso.
Guru: Realizzare un exploit per ogni singolo sorgente vulnerabile pubblicato.

SOLUZIONE CYBER ENIGMA HJ 203

La data di nostro interesse in formato GG/MM era ricavabile a partire dal numero della rivista (203), da cui ottenevamo 20 Marzo che, sommando 10 giorni, diveniva 30 Marzo.
Per l'anno andavano mosse alcune considerazioni relative ai suggerimenti indicati, vediamoli uno per uno:
La data dell'evento di nostro interesse (giorno, mese ed anno) è ben più che in evidenza in questo numero.
La data in formato GG/MM era, come visto, riconducibile al numero della rivista. All'anno ci arriviamo tra poco

Dalla luna alle stelle sono sempre presenti e prime in tutto, malgrado non facciano la dieta sono in perfetto peso forma. Meglio non farle arrabbiare che diventano di fuoco e se in cattiva compagnia anche assassine!

Con questo suggerimento ci riferivamo alle molecole di idrogeno (H), primo elemento chimico della tavola degli elementi, il più abbondante e leggero nell'universo ed altamente infiammabile.
Ma noi siamo fortunati, in questo caso sono tenute per mano dalla chiave della vita, fissandosi l'un l'altra come dinanzi uno specchio insieme ad un amico che spesso è talmente generoso da regalare a tutti diamanti.
Questo ci faceva intuire che l'elemento di nostro interesse oltre che idrogeno dovesse contenere anche Carbonio (C) e che la configurazione molecolare dovesse essere simmetrica e separata dall'ossigeno.
Un'esigenza fisiologica di ogni animale (uomo compreso) rappresenta una buona chiave di lettura per questo Cyber Enigma.
Il sonno!
L'oggetto di discussione ormai è da tempo in disuso.
A questo punto una ricerca su Wikipedia con chiave "30 Marzo" con gli elementi appena evidenziati dai suggerimenti ci doveva far riflettere su questo evento:

1842 - L'anestesia attraverso l'uso dell'etere viene usata per la prima volta in una operazione chirurgica dal dottor Crawford Long (cit. Wikipedia).
L'etere dietilico come anestetico è ormai in disuso da un bel po'... ma osserviamo la sua formula: CH3-CH2-O-CH2-CH3.
Notate nulla ? Configurazione simmetrica e molecole di idrogeno in coppia con il Carbonio.
Prendiamo quindi in considerazione il numero di queste, l'ossigeno come separatore (eliminando quindi le O dal testo crittato) e scriviamo la chiave come: 3223.
A questo punto consideriamo la stringa crittata e la relativa chiave ed applichiamo quindi una traslazione di -n caratteri:

```
F G T Y H N N R Q Q P D Q G U W H V K C C C V R
3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3 3 2 2 3
C E R V E L L O N O N A N E S T E T I Z Z A T O
```

Complimenti a Luca M. da Roma, l'unico ad aver risolto questo Cyber Enigma alla data di stampa del presente numero (5 Luglio 2010). Cosa aspetti ad iscriverti sul nostro forum ?

Inviate tutto a cyberenigma@hackerjournal.it specificando come oggetto della mail il livello (newbie/mid/esperti) ed il numero della rivista del cyberenigma risolto. I migliori saranno pubblicati in questa pagina e sul sito www.hackerjournal.it!

di Massimiliano Rinaldi
redazione@hackerjournal.it

PROGRAMMARE PER IPAD

PROGRAMMING

Il tablet di Apple è stato da poco lanciato sul mercato e con esso i nuovi strumenti di sviluppo. Vediamo come utilizzarli.

Con l'avvento dell'iPad Apple ha dato una rispolverata al suo strumento di sviluppo, l'SDK (Software development kit) giunto alla versione (stabile) 3.2.2 per Snow Leopard.
La versione 3.2.2 può essere scaricata all'indirizzo http://developer.apple.com/iphone/index.action bisogna però sottoscrivere il programma di sviluppo che costa 99 dollari all'anno e che consente di compilare i programmi per distribuirli e venderli sull'Apple Store.
L'idea di sviluppare dei nuovi progetti per iPad potrebbe risultare in questo momento particolarmente conveniente perché lo store italia (e anche quello statunitense del resto) non è così affollato di programmi come lo è lo store dedicato all'iPhone e quindi le possibilità di farsi notare e vendere aumentano sensibilmente.
In questo articolo ad esempio vedremo come sfruttare una delle funzioni più spettacolari dell'iPad, ovvero l'accelerometro che permette il riconoscimento dell'orientamento dell'iPad lungo gli assi X, Y, Z, ovvero in un ambiente tridimensionale, tanto caro agli sviluppatori 3D, in cui muoversi liberamente.
L'accelerometro può essere utilizzato convenientemente nei videogiochi per spostare gli oggetti rilevando l'inclinazione del dispositivo. Tipico l'utilizzo come volante nei giochi di guida. Orientandolo a destra o sinistra si può

programmare un'auto affinché sterzi a destra o a sinistra seguendo proprio l'inclinazione dell'iPad.

AL LAVORO

Per il nostro esempio non partiremo da un progetto così complesso come un videogioco ci limiteremo a programmare una sorta di misuratore del gradimento dei HJ.
Si tratta prevalentemente di un gioco, vogliamo infatti programmare una mini applicazione che, scuotendo l'iPad, visualizzi un numero random compreso tra 1 e 100, naturalmente il tutto condito da un grafica "Hacker Journal".
Prima di tutto lanciamo XCode (è l'applicazione con l'icona a forma di martello che sormonta una A blu stilizzata) selezioniamo il template che fa al caso nostro:
File>New Project>View-based Application

Si tratta di una semplice applicazione che può essere utilizzata con classi del tipo viewController e che personalizzeremo alla "bisogna".

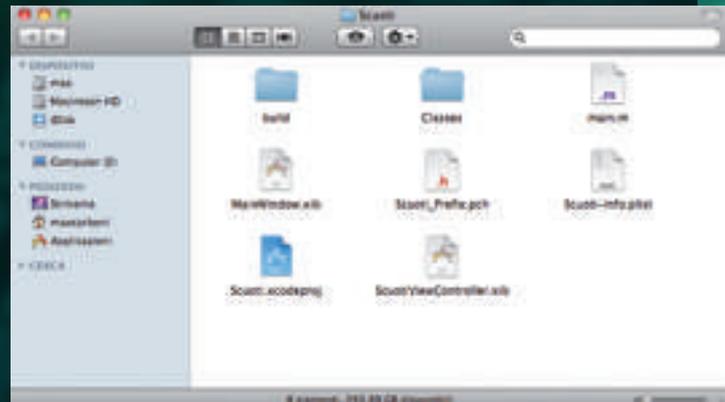
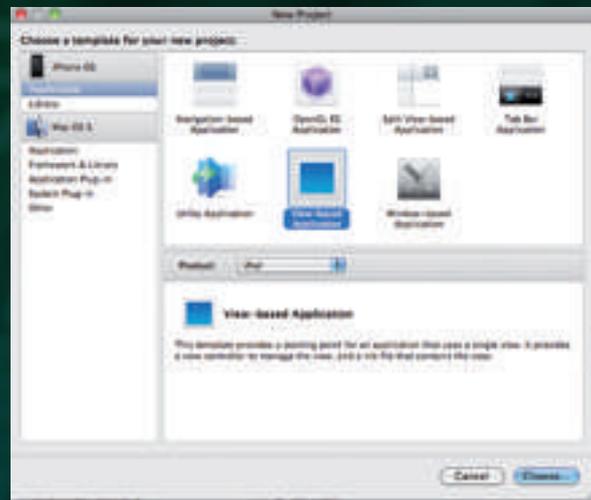
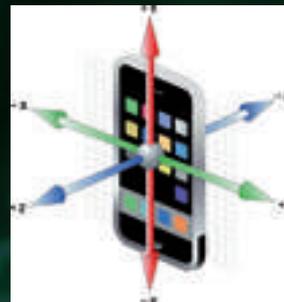
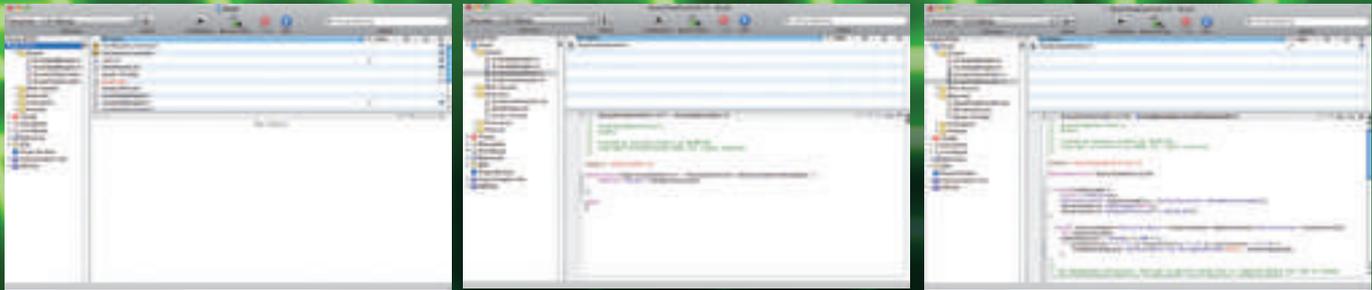
Chiamiamo il nostro progetto con un nome particolarmente evocativo "Scuoti" per fortuna non si tratta di uno dei tanti simulatori di peti già in commercio su Apple Store altrimenti, seguendo la stessa vena ispiratrice, il nome sarebbe stato molto più greve.
In questo modo verrà creata una cartella Scuoti al cui interno trovano spazio tutti gli elementi base da personalizzare.

PROGETTO "SCUOTI"

Lanciamo l'eseguibile Scuoti.xcodeproj che si trova all'interno della cartellina (icona blu con la A stilizzata).
All'interno della finestra che si è aperta, diamo un'occhiata, sulla sinistra nella colonna Groups e Files, agli elementi presenti, in particolare alle classi all'interno della cartella Classes.
Il primo file che andremo a modificare è ScuotiViewController.h, scriviamo il seguente codice modificando in parte quello già esistente:

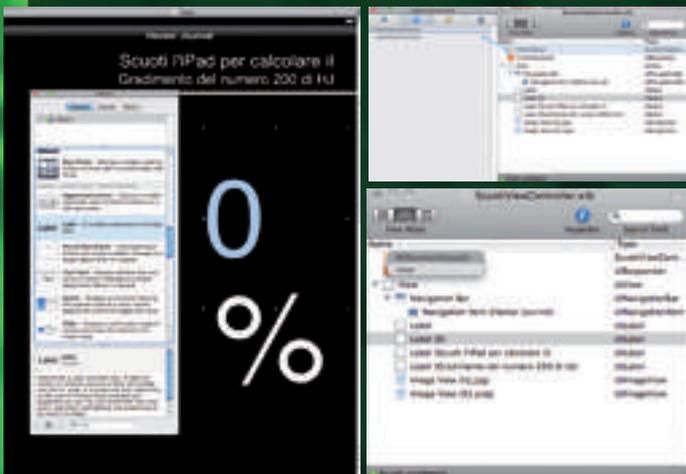
```
#import <UIKit/UIKit.h>

@interface ScuotiViewController : UIViewCon-
troller <UIAccelerometerDelegate> {
       IBOutlet UILabel *lblNumeroCasuale;

}

@end
```

Passiamo quindi al file ScuotiViewController.m, dove aggiungeremo, al codice esistente, queste ulteriori righe di codice:

```
- (void)viewDidLoad {
        [super viewDidLoad];
      UIAccelerometer *accelerometro =
[UIAccelerometer sharedAccelerometer];
       [accelerometro setDelegate:self];
       [accelerometro setUpdateInterval:(1.0f
/10.0f)];
```

```
}
- (void) accelerometer:(UIAccelerometer *)
accelerometer didAccelerate:(UIAcceleration
*)acceleration{
        int numeroCasuale;
        numeroCasuale = random() % 100 + 1;
        if (acceleration.x > 1.5 ||
acceleration.y > 1.5 || acceleration.z >
1.5) {
                [lblNumeroCasuale
setText:[NSString stringWithFormat:@"%i",
numeroCasuale]];
        }
}
```

PERSONALIZZAZIONE DEL CODICE

Come potete notare dal codice, abbiamo impostato il numero casuale da 1 a 100 ma basta agire sulla stringa
numeroCasuale = random() % 100 + 1;
cambiando 100 con 50 per avere un'escursione di numeri casuali che vanno da 1 a 50.
Nella stringa successiva abbiamo scritto che se viene rilevato un movimento maggiore di 1,5 sui tre assi (x, y, z) si attiva la funzione acceleration.
Anche in questo caso la sensibilità di risposta può essere variata semplicemente cambiando il valore 1,5 nella stringa di codice:

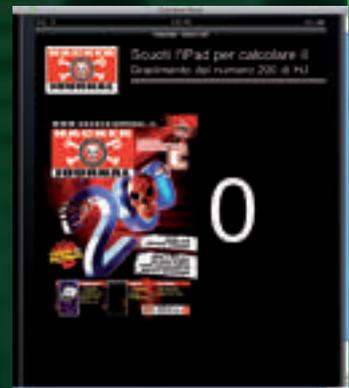
```
if (acceleration.x > 1.5 || acceleration.y >
1.5 || acceleration.z > 1.5) {
```

L'INTERFACCIA

Ora non resta che personalizzare l'interfaccia e collegarla al codice che abbiamo appena scritto per rendere tutte le funzioni (a dire il vero l'unica che abbiamo previsto, ovvero il cambio di numero a seguito di scuotimento) operative.
Nella cartella Resources selezioniamo ScuotiViewController.xib e clicchiamo due volte su di essa per aprire lo strumento di personalizzazione grafica, ovvero Interface Building.
Personalizziamo a piacimento. Nel nostro caso abbiamo inserito il logo di HJ e la copertina semplicemente importando da

Tools>Library

delle Image View, ovvero degli spazi all'interno dei quali visualizzare delle immagini. Per vedere l'immagine all'interno della Image View occorre inserirla nella cartella Scuoti (quella del progetto) e quindi trascinarla all'interno della cartella Resources. A questo punto è selezionabile da:

Tools>Inspector>Images View Attributes>Image

L'elemento fondamentale della nostra applicazione, al di là di tutti gli abbellimenti è la Label che dobbiamo importare nella nostra interfaccia e quindi:
Selezionare Inspector>Label Connections.
Trascinare New Referencing Outlet su File's Owner (come da immagine). Selezionare ibNumeroCasuale (come da immagine). A questo punto l'applicazione è pronta. Laanciamo Build an Run e vediamo il risultato. Una piccola avvertenza, il simulatore di iPad/iPhone, purtroppo non è in grado di simulare gli effetti dell'acceleratore, quindi per accertarvi del corretto funzionamento dell'applicazione occorre caricarla sull'iPad e fare tutte le verifiche del caso.

CODICE SORGENTE

Il file sorgente di questa applicazione può essere scaricato all'indirizzo www.hackerjournal.it nella sezione download.

Ingegneria del software open source

ENGINEERING

L'INGEGNERIA DEL SOFTWARE È UNA DISCIPLINA INFORMATICA CHE STUDIA IL PROCESSO DI SVILUPPO DEL SOFTWARE SECONDO MODELLI PRESTABILITI.

Negli ultimi decenni la Object Oriented Programming (OOP) è diventata il paradigma dominante all'interno del mondo dello sviluppo del software, grazie al suo approccio sistematico che modella la realtà come un insieme organizzato di oggetti. Se fino a poco tempo fa ci si riferiva alla OOP per indicare solamente un paradigma di programmazione, oggi il significato è mutato e descrive l'intero processo di sviluppo del software, abbracciando quindi i problemi studiati dall'ingegneria del software.

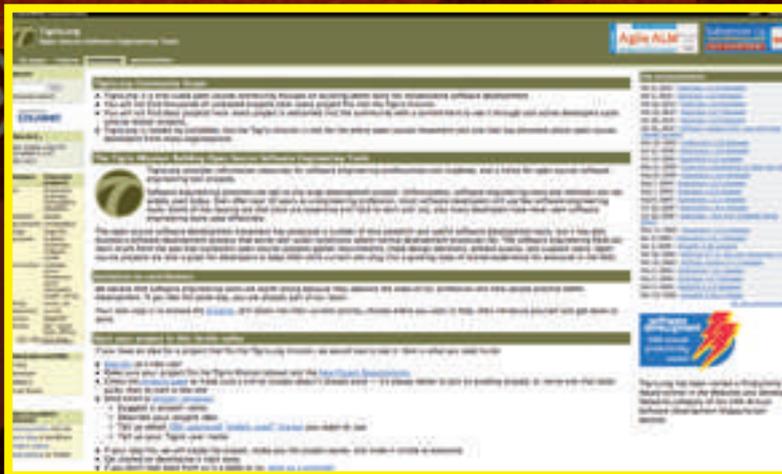
STRATEGIE ORGANIZZATIVE

L'ingegneria del software è una disciplina informatica che studia il processo di sviluppo del software, cercando di determinare valide strategie organizzative per la realizzazione di programmi di qualità. Erroneamente si potrebbe pensare che la creazione di un software consista solamente nella scrittura e nella compilazione di codice; generalmente, invece, il processo che porta alla realizzazione di un software è complesso e si suddivide in numerose attività. Esistono molteplici modelli di processo di sviluppo del software (Waterfall, a spirale, V-Cycle,...) tutti differenti nella struttura, ma simili nelle attività che li compongono.

I MODELLI

Consideriamo un modello semplificato, composto da quattro fasi: Analisi, Progettazione, Implementazione e Verifica.
La fase di Analisi consiste nell'individuazione dei requisiti che il progetto software deve soddisfare; l'analista studia il dominio applicativo al fine di comprendere a fondo il problema che il software dovrà risolvere.
La fase di Progettazione si concentra principalmente su quattro grandi aree: le strutture dati, l'architettura del sistema, le interfacce e i componenti.
L' Implementazione è la fase di scrittura del codice ed è l'attività che generalmente richiede la minor percentuale di lavoro rispetto alle altre tre.
La fase di Verifica consiste nel collaudare il sistema realizzato, al fine di trovare errori e di verificare che gli output siano quelli previsti.

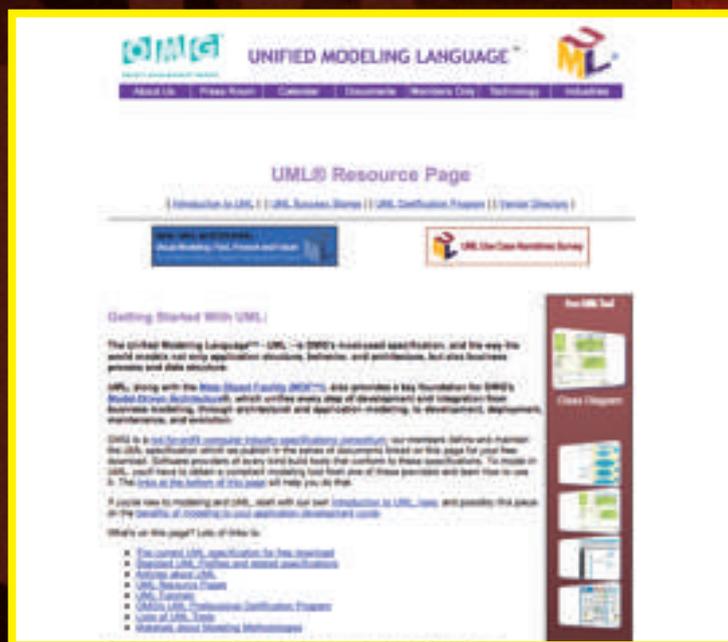
Gli ingegneri del software ricorrono all'utilizzo di strumenti e tecniche che assistono tutte le fasi di creazione del software, a partire dalla modellazione del sistema, per arrivare alla consegna del prodotto finito (ma sul quale è quasi sempre necessario effettuare manutenzione). Questi strumenti sono chiamati tool di C.A.S.E (Computer-Aided Software Engineering) e si differenziano in base all'attività cui fanno da supporto; esistono strumenti per la modellazione del sistema, per la pianificazione del progetto software, per l'analisi dei rischi, per la programmazione, per la gestione della documentazione e tanti altri ancora. Il costo spesso molto elevato di questi strumenti ha portato il mondo Open-Source a produrre una grande quantità di software, alcuni di ottima qualità, per l'assistenza al lavoro dell'ingegnere del software. La filosofia del Free Software ha inoltre contribuito notevolmente al miglioramento di tali

Tigris.org è una comunità di medie dimensioni, sviluppata sui valori dell'open source e incentrata sullo sviluppo di strumenti migliori per lo sviluppo del software collaborativo.

pratiche ingegneristiche, attraverso il consolidamento del concetto di collaborazione: i team di sviluppo non sono più ristretti ad una limitata cerchia di persone, ma gli sviluppatori sono moltissimi e sparsi in tutto il mondo. Questo ultimo tema è stato sinteticamente affrontato da Eric S. Raymond nell'ormai famosissimo scritto "La cattedrale e il bazaar" dove vengono tratte alcune importanti conclusioni, indiscutibilmente a favore del sorgente aperto. Come osservato poco fa, i team di sviluppatori non sono più ristretti e delimitati in fase di pianificazione, non c'è più una gestione piramidale del processo di sviluppo del software; ora gli sviluppatori sono tutti sullo stesso livello, si trovano all'interno di un grande bazaar dove possono fare le loro proposte e dove queste ultime possono essere accettate, modificate, aggiornate. La crescita del numero di programmatori, porta ad un incremento di idee e di novità, le release si susseguono velocemente una dopo l'altra, favorendo il progresso del sistema software.

GLI STRUMENTI

L'approccio OpenSource è quindi qualcosa di radicalmente innovativo ed originale, qualcosa che non segue le classiche regole dell'ingegneria del software. Lo sviluppo di software OpenSource è più rapido ("release early, release often" osserva Raymond) ed è spinto dalla passione e dal divertimento dei programmatori (lo stesso Linus Torvalds ammette di aver iniziato a scrivere il codice del kernel di Linux "just for fun"). Volgendo nuovamente l'attenzione agli strumenti dell'ingegneria del software, il punto di riferimento per il mondo OpenSource è senza dubbio la community Tigris.org (www.tigris.org), che si propone di essere un grande repository di software ed idee legate alla software engineering. Molti sono i progetti attivi hostati, alcuni ancora in fase di studio iniziale, altri già stabili e con un buon numero di release alle spalle; tra i progetti più utilizzati e per questo più noti si segnalano ArgoUML e Subversion.

ARGO UML

ArgoUML è un software che consente di assistere la fase di Analisi del sistema software che si intende realizzare utilizzando il linguaggio di modellazione UML. UML (Unified Modeling Language) è una notazione grafica utilizzata per descrivere le caratteristiche di un progetto software e per assistere le fasi di analisi del sistema. Attraverso il disegno di diagrammi UML è possibile modellare il sistema, cercando di produrne un'ottima astrazione. Non essendo possibile affrontare in questo approfondimento le specifiche del linguaggio UML, i lettori sono invitati a visitare il sito dell'Object Management Group (www.omg.org) e l'homepage www.uml.org. ArgoUML, scritto in Java, offre un ottimo supporto per il disegno di questi diagrammi e inoltre è dotato di qualche funzionalità in più, che fanno di questo software uno strumento indispensabile per l'ingegnere del software OpenSource.

Tra le caratteristiche più evidenti vi è la possibilità di generare automaticamente codice Java (o C++ attraverso l'utilizzo di un plugin) a partire dai Class Diagram (forward engineering) ed è anche possibile effettuare l'operazione contraria di reverse engineering (dai file .class ai diagrammi). Subversion è un software per il controllo delle revisioni che si propone come valido rimpiazzo per il più famoso CVS. Un "version control system" consente di tenere traccia delle modifiche che vengono effettuate ad un file (tipicamente un file sorgente, ma nulla vieta di utilizzarlo per gli scopi più disparati) da uno o più utenti in maniera trasparente e consente di risalire all'interno dell'albero delle modifiche nel momento in cui s'intenda tornare ad una versione precedente di quanto prodotto. Lo standard de facto per i sistemi di controllo delle revisioni è il software CVS (www.cvshome.org), ma sulla sua scia sono apparsi diversi progetti alternativi, che sfruttano la popolarità e il modello di CVS, proponendo soluzioni perfezionate.

SUBVERSION

Uno di questi è Subversion, che è stato progettato, fin dalla prima versione, come sistema collaborativo per utenti remoti; ciò ha consentito di ottenere un ottimo supporto sia dal lato client che dal lato server, eliminando le imprecisioni di CVS, inizialmente pensato per il lavoro in locale. La sintassi dei comandi Subversion è pressoché equivalente a quella in CVS e di conseguenza lo switch tra i due sistemi viene ulteriormente semplificato. Un'altra grande differenza tra i due sistemi consiste nelle possibilità, in Subversion, di effettuare un versionamento direttamente su di un'intera directory, operazione impossibile in CVS; questa feature ha una grande rilevanza in quanto consente di ottenere un approccio maggiormente modulare allo sviluppo del sistema.

LE GUI

Come per CVS, anche per Subversion è disponibile una grande quantità di GUI che permette di gestire il versionamento con pochi clic del mouse. Ai neofiti non consigliamo comunque di affidarsi alle interfacce grafiche, solo apparentemente più intuitive; esse sono spesso caotiche e colme di comandi. Sia CVS che Subversion possono essere utilizzati produttivamente da linea di comando, anche conoscendo i pochissimi comandi principali.

Giovanni `m0le`Federico - giovanni@m0le.it
Fabio `BlackLight` Manganiello - blacklight@autistici.org

PARTE IV

CORSO DI PROGRAMMAZIONE IN C

LINGUAGGI La gestione della memoria è un aspetto cruciale. Disporre di strumenti in grado di manipolare ed indicizzare i dati trattati dal calcolatore in memoria costituisce una conoscenza di primaria importanza per il neoprogrammatore.

L'intera quarta parte di questo corso sarà pertanto destinata a ciò, introducendo un tipo di dato nuovo, di fondamentale rilievo per tutto il nostro percorso: stiamo parlando, naturalmente, dei puntatori. Non ci stancheremo mai di ricordarvi che i luoghi ideali per gli approfondimenti legati a questo corso di programmazione restano il forum della rivista raggiungibile all'indirizzo www.hackerjournal.it ed il canale #hackerjournal su irc.azzurra.org. Fatevi sentire!

PUNTATORI

Inauguriamo l'inizio di questa quarta parte del corso di programmazione in C con la seguente, importantissima:

DEFINIZIONE 23
PUNTATORE

Definiamo ed identifichiamo con il nome **"puntatore"** una variabile il cui valore è un indirizzo di memoria.
La dichiarazione di un puntatore avviene attraverso l'utilizzo di un apposito operatore unario rappresentato dal carattere asterisco (*) nella forma **<tipo> *<puntatore>**.

Evitando di percorrere una strada piena di inutili formalismi risulterà sicuramente più semplice comprendere concettualmente il significato dei puntatori ricordandosi cosa accade in memoria nel momento in cui si inizializza un determinato valore. Sembrerà difatti lapalissiano asserire che quando si effettua una dichiarazione del tipo "int n = 10" si sta assegnando ad una ben specifica locazione di memoria il valore "10" (di tipo intero). Appare lecito quindi domandarsi, ora che gran parte dei concetti di base della programmazione in C sono chiari, se esiste un modo attraverso il quale riferirsi a quella specifica zona di memoria. Una risposta esauriente e funzionale la si ritrova nell'utilizzo dei puntatori; in altri termini (ma non gli unici) essi offrono un meccanismo decisamente flessibile per trattare indirizzi di memoria sotto-forma di "etichette" (e quindi variabili) e non solo. Comprendere ed usare correttamente i puntatori costituisce non solo l'obiettivo di questa quarta parte, bensì un attributo cruciale che caratterizza il linguaggio stesso e le possibilità applicative. Pur non sapendolo, abbiamo già incontrato puntatori quando abbiamo parlato di vettori e di funzioni (vedremo tra poco perché) e li incontreremo nuovamente quando introdurremo le strutture. Consideriamo il seguente esempio attraverso il quale potremo capire meglio il concetto di puntatore:

```
#include <stdio.h>
int main(){
      int n = 10;
      int *ptr = &n;
      fprintf(stdout, "Il valore di n: %d\n", n);
      fprintf(stdout, "Indirizzo di n: %p\n", &n);
      fprintf(stdout, "Il valore di ptr: %p\n", ptr);
      fprintf(stdout, "Indirizzo di ptr: %p\n",
&ptr);
      return 0;
}
```

Compiliamo ed osserviamo l'output:

```
$ gcc point.c -o point
$ ./point
Il valore di n: 10
Indirizzo di n: 0x7fff1254e2ec
Il valore di ptr: 0x7fff1254e2ec
Indirizzo di ptr: 0x7fff1254e2e0
```

Con questo elementare sorgente abbiamo semplicemente collocato in memoria due variabili: la prima è "n", di tipo intero e con valore 10, la seconda è un puntatore a quest'ultima. Osserviamo il comportamento dello stesso nelle righe dell'output evidenziate. L'indirizzo della variabile "n" in memoria è **0x7fff1254e2ec** mentre il suo valore, come ci aspettavamo avendolo assegnato a mano, è "10". La situazione per quanto concerne il puntatore è invece diversa: notiamo che esso possiede un indirizzo proprio **(0x7fff1254e2e0)** ma il valore di quest'ultimo è esattamente l'indirizzo di memoria della variabile puntata (n), ovvero, di nuovo, **0x7fff1254e2ec**. Questo comportamento è del resto quello che prevedevamo in base alla definizione pocanzi data di puntatore.
A questo punto, avendo memorizzato all'interno della variabile "ptr" l'indirizzo di memoria della variabile "n", possiamo accedere al suo valore utilizzando **l'operatore di indirezione o dereferimento (*)**. Espandiamo pertanto il sorgente analizzato ed osserviamo il suo comportamento durante l'esecuzione:

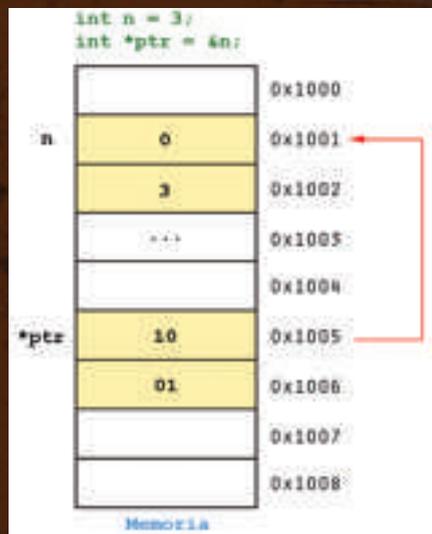
```
#include <stdio.h>
```

```
int main(){
	int n = 10;
	int *ptr = &n;
	fprintf(stdout, "Il valore di n: %d\n", n);
	fprintf(stdout, "Indirizzo di n: %p\n", &n);
	fprintf(stdout, "Il valore di ptr: %p\n", ptr);
	fprintf(stdout, "Indirizzo di ptr: %p\n",
&ptr);
	fprintf(stdout, "Valore puntato da ptr: ¬
%d\n", *ptr);
	return 0;
}
```

Da cui l'output sarà:

```
$ ./point
Il valore di n: 10
Indirizzo di n: 0x7fff31bc32ec
Il valore di ptr: 0x7fff31bc32ec
Indirizzo di ptr: 0x7fff31bc32e0
Valore puntato da ptr: 10
```

Intuire il meccanismo di azione di quanto appena espresso è facile. Riferendoci alla cella di memoria contenente il valore di "n" attraverso il suo puntatore abbiamo stampato a video il contenuto della stessa. Semplice, no ?

Dichiarazione di un puntatore a intero

ARITMETICA DEI PUNTATORI

Abbiamo già detto nel corso della terza parte che la dichiarazione di un array in C si traduce in un puntatore al primo elemento del vettore. Da qui pertanto riprendiamo in considerazione per un attimo il sorgente con il quale ci siamo lasciati nel numero precedente:

```
1. #include <stdio.h>
2. #include <stdlib.h>

3. int main()
4. {
5.		int i;
6.		int *miovettore;
7.		miovettore = (int*) ¬
malloc(10*sizeof(int));

8.		for (i=0; i < 10; i++) {
9.				fprintf (stdout, "Elemento ¬
n.%d: ", i+1);
10.				scanf ("%d", ¬
&miovettore[i]);
11.		}

12.		for (i=0; i < 10; i++)
13.				fprintf (stdout, "L'eleme ¬
nto n.%d vale %d\n", i+1, miovettore[i]);
14. }
```

Di nostro interesse sono le righe 6 e 7 nelle quali abbiamo dichiarato un puntatore di tipo intero "miovettore" ed a questo abbiamo riservato attraverso l'apposita funzione **malloc()**, che analizzeremo tra pochissimo, uno spazio di memoria idoneo a contenere 10 valori di tipo intero, moltiplicando quest'ultimo numero per la dimensione di un intero attraverso la funzione **sizeof()** (anche questa oggetto di analisi a breve). Sostanzialmente abbiamo definito un **array** in modo diverso dalla classica assegnazione vista nella terza parte del corso. Ci aspettiamo, per quanto detto allora e ripetuto in queste pagine, che le 10 locazioni di memoria assegnate contengano ognuna un elemento del vettore, così come espresso dal ciclo che le stampa tutte come per qualsiasi altro array alle righe 12, 13 e 14. Proviamo quindi a riscrivere tutto il programma lavorando **direttamente sugli indirizzi di memoria** utilizzando le apposite operazioni aritmetiche spendibili sui puntatori. Il sorgente nuovo sarà quindi:

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main()
4.  {
5.	int i;
6.	int *miovettore;
7.	miovettore = (int*) malloc(10*sizeof(int));
8.	for (i=0; i < 10; i++) {
9.		fprintf (stdout, "Elemento n.%d:
",¬ i+1);
10.		scanf ("%d", (miovettore + i));
11.	}

12.	for (i=0; i < 10; i++)
13.		fprintf (stdout, "L'elemento n.%d ¬
vale %d\n", i+1, *(miovettore+i));
14. }
```

Prestiamo particolare attenzione alle righe 10 e 13 (evidenziate).

Nella prima abbiamo sostituito l'istruzione **scanf ("%d", &miovettore[i]);** con **scanf ("%d", (miovettore + i));**.
Il motivo è semplice. Ricordandoci che un array, come più volte detto, non è altro che un puntatore riferito alla prima locazione del "blocco di memoria" che dovrà contenere tutti i valori dello stesso analizziamo passo per passo il comportamento della riga 10 tenendo bene a mente quanto espresso alla riga 7. Torniamo all'esempio della cassettiera visto nello scorso numero ed immaginiamo la riga 7 come un "falegname" al quale stiamo chiedendo di costruire una cassettiera con 10 tiretti. Il primo di questi tiretti sarà quello referenziato dal puntatore "miovettore".
Alla riga 10, a prima vista, si potrebbe pensare di star sommando un intero (i è una variabile contatore del ciclo for definito alla riga 8 che va da 0 a 9) ad un puntatore e quindi ad un indirizzo di memoria. Mai nulla di tanto errato!
In questo caso, infatti, assistiamo alla più concreta testimonianza della capacità di controllo della memoria da parte del C attraverso banali operazioni aritmetiche eseguite direttamente nella memoria del Calcolatore. La scrittura **(miovettore + i)** sta infatti a significare "avanza di i posizioni rispetto all'indirizzo di partenza referenziato dal puntatore". Essendo un vettore di interi ogni "posizione" costituirà 4 byte, pertanto: "miovettore + 0" si riferirà alla prima posizione dell'array entro cui andremo a scrivere il valore preso da tastiera attraverso la **scanf()** (che ricordiamo essere una funzione **insicura**, il perché lo lasciamo scoprire a voi), "miovettore + 1" si riferirà all'indirizzo di "miovettore" più quattro byte (e quindi alla seconda posizione dell'array, il secondo cassetto per intenderci), "miovettore + 2" sarà "miovettore" + otto byte (terza posizione, terzo cassetto) e via di seguito.
Alla riga 13 **accediamo** agli elementi del vettore con la medesima aritmetica, stavolta anteponendo l'operatore di indirezione visto prima attraverso il quale indichiamo di volerci riferire al valore dell'indirizzo di memoria contenuto dal puntatore e non all'indirizzo stesso.
Compiliamo ed eseguiamo il nuovo programma per testare che, effettivamente, l'output ed il funzionamento dello stesso risulti immutato:

```
$ gcc point2.c -o point2
$ ./point2
Elemento n.1: 1
Elemento n.2: 2
...
Elemento n.10: 10
L'elemento n.1 vale 1
L'elemento n.2 vale 2
```

```
...
L'elemento n.10 vale 10
```

Per rendere maggiormente chiaro e verificato quanto appena espresso, proponiamo un'ulteriore riscrittura del programma modificando la riga 13 da **fprintf (stdout, "L'elemento n.%d vale %d\n", i+1, *(miovettore+i)); a fprintf (stdout, "Indirizzo #%d elemento: %p\n", i+1, (miovettore+i));**.
Osserviamo con attenzione l'output:

```
$ ./point3
Elemento n.1: 1
Elemento n.2: 2
Elemento n.3: 3
...
Elemento n.10: 10
Indirizzo #1 elemento: 0x601060
Indirizzo #2 elemento: 0x601064
Indirizzo #3 elemento: 0x601068
Indirizzo #4 elemento: 0x60106c
Indirizzo #5 elemento: 0x601070
Indirizzo #6 elemento: 0x601074
Indirizzo #7 elemento: 0x601078
Indirizzo #8 elemento: 0x60107c
Indirizzo #9 elemento: 0x601080
Indirizzo #10 elemento: 0x601084
```

Notate nulla ? Il distacco tra un indirizzo relativo ad un elemento e l'altro è di esattamente 4 byte, come volevasi dimostrare. Dovrebbe essere ora maggiormente chiaro per il lettore "cosa accade" in memoria all'atto di dichiarazione ed inizializzazione di un array e di una variabile, ma questi dettagli potevamo scoprirli solo una volta introdotti i puntatori... tutto sommato, inizia a tornare utile conoscerli, no ?

PUNTATORI A PUNTATORI

La domanda appare quasi scontata arrivati a questo punto del corso: è possibile puntare ad un puntatore ? La risposta è naturalmente affermativa ed abbastanza intuitiva. Consideriamo il seguente sorgente esemplificativo ed osserviamo l'output dopo averlo compilato:

```
#include <stdio.h>
int main(){
	int n = 203;
	int *p1 = &n;
	int **p2 = &p1;
	fprintf(stdout, "Il numero di questa rivista e' il %d (senza puntatore)\n", n);
	fprintf(stdout, "Il numero di questa rivista e' il %d (con puntatore a n)\n", *p1);
	fprintf(stdout, "Il numero di questa rivista e' il %d (con puntatore a p1)\n\n", **p2);

	fprintf(stdout, "Indirizzo di n: %p\n", &n);
	fprintf(stdout, "Indirizzo di p1: %p\n", &p1);
	fprintf(stdout, "Indirizzo di p2: %p\n\n", ¬
&p2);

	fprintf(stdout, "Valore di p1: %p\n", p1);
	fprintf(stdout, "Valore di p2: %p\n", p2);

	return 0;
}
```

```
$ gcc ptop.c -o ptop
$ ./ptop
Il numero di questa rivista e' il 203 (senza puntatore)
Il numero di questa rivista e' il 203 (con puntatore a n)
Il numero di questa rivista e' il 203 (con puntatore a p1)

Indirizzo di n: 0x7fffc37242ec
Indirizzo di p1: 0x7fffc37242e0
Indirizzo di p2: 0x7fffc37242d8

Valore di p1: 0x7fffc37242ec
Valore di p2: 0x7fffc37242e0
```

Per quanto elementare, questo sorgente ha una notevole rilevanza didattica. Il valore del puntatore "p2" è l'indirizzo del puntatore "p1". Riferendoci a "p2", quindi, identifichiamo non il valore di "p1" bensì **l'indirizzo proprio** del puntatore "p1". Utilizzando una **doppia indirezione (**)** il meccanismo è quindi lo stesso applicato finora ai puntatori ma ripetuto due volte. In prima istanza interroghiamo l'indirizzo referenziato da "p2" (il suo valore), attraverso questo "saltiamo" a quell'indirizzo dove ad aspettarci è un ulteriore puntatore il cui valore si riferirà ad una zona distinta di memoria (quella di n). In definitiva:
1. Accediamo in memoria all'indirizzo **0x7fffc37242d8** (quello di **p2**).
2. Leggiamo il valore contenuto dal puntatore **p2** in questa zona (**0x7fffc37242e0**).
3. Accediamo in memoria all'indirizzo **0x7fffc37242e0**.
4. Leggiamo il valore contenuto in questa zona che scopriamo essere **p1** (**0x7fffc37242ec**).
5. Accediamo in memoria all'indirizzo **0x7fffc37242ec**.
6. Leggiamo il valore presente in questa zona: **203**. Torneremo sui puntatori a puntatori a breve in quanto questi rappresentano la modalità attraverso la quale è possibile gestire il passaggio di puntatori a parametri nelle funzioni. In questa sede deve essere necessariamente detto che è inoltre possibile, attraverso i puntatori, riferirsi a strutture dati che ne contengono altri al loro interno, ma per questo il lettore dovrebbe conoscere cosa siano le strutture dati e come manipolarle, concetti abbastanza avanzati che, non potendo brevemente esplicare in queste pagine, demandiamo alle parti conclusive del corso.

PUNTATORI SENZA TIPO

Spesso può essere utile utilizzare puntatori senza tipo, contenenti ovvero un indirizzo di memoria generico, non legato ad una particolare tipologia di dato. Per dichiararli la sintassi è void *puntatore;.

Vediamo un semplice esempio:

```
#include <stdio.h>
int main(){
	int n = 44;
	void *ptr = &n;
	int *ptr2 = ptr;

	fprintf(stdout, "Valore di ptr:\t %p\n", ptr);
	fprintf(stdout, "Valore di ptr2:\t %p\n",
ptr2);
	return 0;
}
```

```
$ gcc point4.c -o point4
$ ./point4
Valore di ptr:   0x7fff33a6d2dc
Valore di ptr2:  0x7fff33a6d2dc
```

Come visibile dall'output dell'applicativo, l'indirizzo restituito dal puntatore non cambia, indipendentemente dal tipo utilizzato. Torneremo a breve su questo aspetto illustrando come e dove può essere utile adottare simili meccanismi.

ALLOCAZIONE DINAMICA DELLA MEMORIA

Usando i puntatori è possibile allocare dinamicamente dello spazio in memoria.
Questa è una possibilità molto utile, ricordando

che finora abbiamo imparato solo come allocare della memoria staticamente (al momento della dichiarazione di una variabile diciamo implicitamente quanto è grande, ad esempio una variabile int o char avranno la stessa dimensione prefissata sulla macchina su cui andiamo a compilare il codice, mentre quando dichiariamo un array dichiariamo anche la sua dimensione). Non è comunque sempre possibile sapere in anticipo quanto spazio in memoria si vuole allocare nel proprio programma. Si pensi ad esempio a un software che richiede l'immissione di dati in una quantità dipendente da una variabile inserita dall'utente o una funzione che legge un'intera riga da file o da standard input senza conoscerne a priori la dimensione.
In casi come questi l'uso di array statici fa perdere flessibilità al programma, o peggio può causare problemi di overflow nel caso in cui la quantità di dati letta sia maggiore di quella prevista nel buffer. Si usa allora in questi casi la cosiddetta allocazione **dinamica**, prevista in C dalla funzione **malloc()** e dalla sua duale **free()**.
La funzione malloc() prende come parametro la quantità di memoria che si vuole allocare (N.B. in byte) e ritorna un puntatore a **void*** che può essere salvato nel puntatore di proprio interesse. Nel caso in cui la quantità di memoria richiesta non può essere allocata, la malloc() ritorna NULL. È sempre buona norma controllare il valore di ritorno di malloc() per assicurarsi che non ci siano stati errori e non compromettere in questo modo la stabilità del programma usando puntatori non correttamente allocati.
Il seguente codice richiede all'utente quante variabili intere vuole inserire e ritorna un array allocato dinamicamente via malloc() contenente quel numero di elementi:

```
int n, *array;

printf ("Numero di variabili che l'array dovra'
contenere: ");
scanf ("%d", &n);

if ((array = (int*) malloc( n*sizeof(int) )) == NULL)
  printf ("C'e' stato un errore irreversibile nell'al-
locazione");
else
  printf ("Allocazione avvenuta con successo");
```

Si noti innanzitutto che per allocare n variabili intere la malloc() vuole come argomento n * sizeof(int). Questo perché l'argomento della malloc() deve essere il numero di byte da allocare, e noi vogliamo che siano allocati n byte moltiplicati per la dimensione in byte di una variabile int (generalmente 4 byte).

Altra cosa da notare è il cast esplicito a int* usato per il valore di ritorno in quanto malloc() ritorna un puntatore alla memoria allocata come void*, ovvero come tipo generico che va poi "specializzato" attraverso un operatore di cast (Puntatori senza Tipo). La maggior parte dei compilatori C non obietterà se questo cast non dovesse essere utilizzato, ma i compilatori C++ sono più restrittivi su queste cose, considerano ambiguo il cast implicito da void* e ritorneranno un errore. Per completezza è quindi sempre buona norma "castare" il valore di ritorno di malloc(). Una volta allocato in questo modo, il vettore può essere acceduto come si farebbe con un normale vettore statico:

```
for (i=0; i < n; i++) {
  printf ("Valore %d: ", i+1);
  scanf ("%d", &array[i]);
}
```

Un'altra funzione molto utile nell'ambito dell'allocazione dinamica della memoria è **realloc()**. Tale funzione consente di modificare la dimensione di un'area di memoria allocata al volo, prende come parametri il puntatore all'area di memoria da modificare e la sua nuova dimensione. Così come la malloc() ritorna un puntatore a void* che punta alla nuova area di memoria.

```
int  i, value, size = 0;
int* array = NULL;

do  {
  printf ("Inserisci un nuovo valore nell'array "
        "(-1 per terminare): ");
  scanf ("%d", &value);

  if (value == -1)
    break;  // Esce dal ciclo

  if ((array = (int*) realloc( array, (++size)* ¬
sizeof(int) )) == NULL)  {
    printf ("Problema fatale nella riallocazione,
¬ esco\n");
    exit(-1);
  }

  array[size-1] = value;
} while (value != -1);

printf ("Sono stati inseriti %d elementi:\n",
size);

for (i=0; i < size; i++)
  printf ("%d\n", array[i]);

free(array);
```

RICHIAMO TEORICO 7

MALLOC() ED HEAP

Due paroline sul modo in cui opera malloc(). Le variabili che abbiamo imparato a dichiarare finora, semplici o vettoriali, vengono allocate a basso livello su una zona di memoria chiamata stack (o sul segmento data nel caso di variabili globali). Ogni funzione ha il suo stack, quindi il main() avrà il suo e ogni funzione dichiarata nel programma avrà uno stack diverso da quello del main(). Lo stack di una funzione viene creato al momento in cui la funzione è richiamata, sono piazzate dentro tutte le variabili e gli array dichiarati all'inizio della funzione, e viene distrutto (deallocato dalla memoria) quando la funzione termina.
Usando la malloc() invece la memoria viene allocata in una zona chiamata heap, generalmente molto grande (proprio per permettere l'allocazione dinamica senza problemi) e condivisa da tutte le funzioni del programma. Questo vuol dire che, se tale spazio in memoria non viene esplicitamente deallocato, rimane lì marcato come "occupato" anche quando la funzione che l'ha allocato è terminata, oppure lo spazio stesso non serve più.

Si noti che la malloc() in questo spezzone di codice non viene mai richiamata, mentre invece alla prima chiamata del ciclo la realloc() viene chiamata su array che è ancora un puntatore a NULL. Questo è un comportamento molto interessante di realloc(): se viene richiamata su un puntatore a NULL non ancora allocato, infatti, alloca automaticamente quella zona di memoria, operando come una comune malloc().

NOTA
Tutte le funzioni prese in esame (malloc(), realloc(), free()) si possono usare a patto di includere l'header stdlib.h all'inizio del codice (#include <stdlib.h>).

PUNTATORI E FUNZIONI

L'uso dei puntatori è indispensabile qualora si vogliano rendere le modifiche operate a variabili passate come argomento di una funzione ope-

rative al di fuori della funzione stessa.
Si pensi ad esempio a una funzione che, presi come argomenti due variabili intere, scambi i valori contenuti in esse (ovvero il valore di a diventa il valore di b e viceversa). Una prima implementazione che viene in mente è la seguente (usando una variabile di appoggio):

```
void swap (int a, int b) {
  int tmp = a;
  a = b;
  b = tmp;
}
```

Una volta invocata questa funzione, ci ritroveremo di fronte ad una sorpresa: lo scambio, materialmente, non viene effettuato al di fuori della funzione. Questo perché abbiamo passato a e b per valore, ovvero abbiamo passato alla funzione i valori di a e b, non le effettive locazioni di memoria che contengono queste variabili. Questo significa che quando la funzione viene richiamata prende i valori che le sono stati passati, li copia in due variabili locali chiamate a e b (che, N.B., sono due variabili diverse da quelle della funzione chiamante, in quanto vengono create sullo stack della nuova funzione e hanno visibilità locale alla funzione stessa) ed effettua le operazioni richieste su queste variabili locali. Una volta che la funzione è terminata, anche il suo stack viene distrutto, e con quello vanno perse anche le modifiche effettuate sulle variabili. Se vogliamo effettivamente operare lo scambio e fare in modo che sia visibile anche alla funzione chiamante, la via è quella di passare le due variabili non per valore, ma per riferimento, ovvero passare alla funzione i puntatori che identificano le zone di memoria che contengono le variabili. La funzione modificherà così direttamente quelle zone di memoria, non copie locali delle variabili, ed i cambiamenti saranno visibili anche al di fuori della funzione:

```
void swap (int* a, int* b) {
  int tmp = *a;  // tmp contiene il valore puntato da a
  *a = *b;  // Il valore puntato da a è uguale al valore puntato da b
  *b = tmp;  // Il valore puntato da b è uguale al vecchio valore di a
}
```

La funzione verrà richiamata nel seguente modo:

```
int a = 2, b = 3;
...
swap (&a, &b);
```

RICHIAMO TEORICO 8

MEMORY LEAK

L'utilizzo della memoria heap può dar luogo a problemi molto seri (e spesso anche molto difficili da scovare), specie in grossi progetti, noti come memory leak. Il principio di base è che la memoria allocata va poi sempre deallocata. Può capitare di scrivere un enorme loop in cui si alloca della memoria dinamicamente, per poi dimenticarsi di deallocare magari un solo byte di quella memoria. Quel loop però legge magari dati da un enorme database venendo quindi eseguito anche un miliardo di volte. Un byte non deallocato moltiplicato per un miliardo di iterazioni fa quasi un gigabyte di memoria RAM non più usata e marcata come "occupata", e questo rischia di influire drammaticamente sulle prestazioni sia del proprio software, sia della macchina su cui si fa girare. Problemi come questi non sono neanche facili da scovare. Ci sono tool appositi come valgrind che aiutano il programmatore ad individuare eventuali leak, ma anche così nella maggior parte dei casi il programmatore, una volta lanciato il suo bel programma magari di 10000 righe di codice, vede la dimensione della memoria occupata aumentare drammaticamente, e non riesce generalmente a scovare, se non dopo ore o giorni di debug, la fonte del problema. È per questo che bisogna stare attenti a deallocare sempre tutta la memoria che si alloca dinamicamente, usando la funzione free() (nel caso di sopra, una volta che lo spazio occupato da array non ci serve più chiameremo free(array)). Il trucco sta nel piazzare subito una free() dopo la malloc() per evitare di dimenticarsela in seguito e scrivere il codice nel mezzo fra le due chiamate.

Si noti che a e b vengono passati per riferimento (la & davanti), a identificare che si vogliono passare non i loro valori ma i loro indirizzi in memoria. Scritture come queste sono utili ogni qualvolta si vuole che una funzione ritorni più di un valore. Per definizione, infatti, una funzione in C ritorna un solo valore. Come fare quindi se si vuole scrivere una funzione che muova il cursore sullo schermo a una posizione (x,y), ritorni al chiamante le coordinate della nuova posizione (come due variabili intere) e ritorni 0 in caso di successo e -1 in caso di errore? Una via può essere quella di ritornare un vettore contenente come primo elemento il successo o meno dell'operazione e come rimanenti due valori le nuove coordinate, ma è un approccio decisamente poco elegante in quanto ritorna un'entità in memoria che, pur contenendo variabili dello stesso tipo strettamente parlando, raggruppa oggetti che sono logicamente non correlati tra loro. La soluzione migliore è quindi quella di passare x e y per riferimento alla funzione e fare in modo che la funzione ritorni un intero che è il "codice di successo" della funzione, qualcosa del tipo:

```
int moveToXY (int* x, int* y)  { /* codice */ }
...
int ret, x, y;
ret = moveToXY(&x, &y);
```

Un altro problema molto comune è come fare in modo che una funzione ritorni un array. Uno è quello di passare l'array alla funzione come puntatore. Dovrebbe ormai essere chiaro che passando un array ad una funzione non facciamo altro che passare il puntatore al primo elemento e tutte le modifiche effettuate su di esso e sugli elementi in memoria successivi saranno visibili anche al chiamante. Vediamo ad esempio una funzione che riempie un array di n elementi

```
void fillArray ( int *v, int size ) {
  int i;
  for (i=0; i < size; i++) {
    printf ("Element n.%d: ", i+1);
    scanf ("%d", &v[i]);
  }
}
...
int v[3];
fillArray(v,3);
```

Un altro modo è quello di ritornare direttamente l'array come puntatore, ma attenzione a scritture del genere:

```
int* foo()  {
  int v[10];
  ...
  return v;
}
...
int *v = foo();
```

RICHIAMO TEORICO 9

MEMORY LEAK: RIMEDI

Linguaggi più avanzati come Java implementano nativamente un meccanismo noto come garbage collection, ovvero algoritmi che operano periodicamente sulla memoria del processo e deallocano automaticamente aree non più utilizzate (ma è anche vero che tali linguaggi non concedono la stessa libertà al programmatore sui processi di allocazione e deallocazione della memoria).
Qualcosa di molto simile è possibile anche in C usando la libreria nota come libgc, e usando invece delle funzioni malloc() e free() della libreria standard la funzione GC_malloc() specificata al suo interno, ma tale libreria è usata relativamente, in quanto si trasformerebbe in molti casi in una dipendenza aggiuntiva per il programma, che potrebbe limitarne la sua esecuzione su alcune macchine.

Scrivendo una cosa del genere il compilatore (nel nostro caso, ricordiamo, gcc o derivati) darà il seguente warning:

```
warning: function returns address of local variable
```

Il motivo, arrivati a questo punto, dovrebbe essere ovvio. Abbiamo dichiarato v all'interno di foo() come array statico di 10 elementi, quindi v è un oggetto locale a foo() allocato sullo stack della funzione stessa al momento della sua creazione e, quindi, distrutto quando la funzione termina. Se eseguissimo quel codice e provassimo a leggere dalla funzione chiamante gli elementi contenuti dentro v ci ritroveremmo a leggere valori più o meno casuali o nulli. Questo perché stiamo leggendo da una zona di memoria che è stata deallocata. Il modo per ritornare un array da una funzione è quello di allocare dinamicamente l'array via malloc() o realloc(). Si è già detto infatti che tali funzioni allocano dinamicamente lo spazio sullo heap, che è una zona di memoria che non viene deallocata finché il processo rimane in vita a differenza dello stack, quindi quello che c'è lì rimane visibile anche al chiamante. Ma essendo il vettore allocato dinamicamente bisogna sempre ricordarsi di deallocarlo dal chiamante usando la free() quando non serve più:

```
    int* foo() {
  int *v = (int*) malloc(10*sizeof(int));
  ...
  return v;
}
...
int *v = foo();
...
free(v);
```

PUNTATORI A FUNZIONI

Una caratteristica estremamente potente del C è quella di poter accedere all'indirizzo di memoria di una funzione oltre che di una variabile o di un vettore, attraverso i puntatori, e di passare una funzione ad un'altra funzione come parametro o di fare in modo che una funzione ritorni a sua volta un puntatore a funzione, esattamente come si farebbe con una variabile intera o un vettore di char. Diamo pertanto la seguente:

DEFINIZIONE 24
PUNTATORI A FUNZIONI

La scrittura per dichiarare in C un puntatore a funzione è tipo (*nome_puntatore)(argomenti).

Questo meccanismo garantisce al programma una flessibilità ed una modularità unica se sfruttata bene. Si pensi ad esempio all'interazione fra HTML e JavaScript in un form del tipo onClick='funzioneJavaScript()'. A basso livello questa scrittura viene gestita come un puntatore a funzione, ovvero qualcosa del tipo se l'evento "click" è vero, allora richiama la funzione puntata dall'identificatore "funzioneJavaScript".

```
int foo (int n) {
  return n;
}
...
int (*p)(int) = foo;
int n = p(3);
```

Si noti che in questo piccolo esempio abbiamo dichiarato una funzione chiamata foo che fa ben poco (prende come parametro un intero n e lo ritorna), quindi un puntatore a funzione p che è di tipo intero (ovvero ritorna un intero) e prende come parametro un int, e tale puntatore contiene l'indirizzo della funzione foo.
A questo punto, possiamo richiamare il puntatore foo esattamente come se richiamassimo la funzione foo. Se usati bene, i puntatori a funzione consentono di risparmiare molto codice e di guadagnare molto in modularità.
Si pensi a un esempio di algoritmo molto comune nell'intelligenza artificiale, che consiste nel verificare se un insieme di vincoli su un insieme di dati è verificato e calcola automaticamente i nuovi domini delle variabili in funzione dei vincoli forniti. Immaginiamo che tali operazioni vengano eseguite all'interno di una funzione.
Rimane il problema di come passare i vincoli. Un metodo può essere quello di "hard-codarli" all'interno della funzione stessa, ma è un metodo decisamente poco flessibile, in quanto se cambiano i vincoli dovremo cambiare il codice della funzione principale stessa.
Un metodo più flessibile è quello di dichiarare i vincoli in una funzione separata e passare tale funzione a quella contenente la "logica" vera e propria dell'algoritmo come puntatore a funzione.
In tal modo, se cambiano i vincoli dobbiamo solo cambiare la funzione da passare alla funzione contenente la "logica", non la funzione stessa. Se, ad esempio, il nostro insieme di dati è un vettore di 4 interi e il vincolo è che il primo e il terzo elemento siano diversi, avremo una funzione del tipo

```
int constraint (int *v) {
  return (v[0] != v[2]);
}
...
int logic (int *v, int (*c)(int*)) {
  if (c(v) == 0) {
    printf ("Il vincolo non e' rispettato, la funzione termina");
    return -1;
  } else {
    // Codice
  }
}
...
logic (v, constraint);
```

Se volessimo modificare il vincolo di integrità, cambieremo semplicemente la funzione da passare come parametro alla funzione logic(), senza toccare il codice della funzione logic() stessa.

Lo *SCANDALO* che ha scosso il *MONDO!*

VATICANO
PEDOFILIA
LE CARTE, LE STORIE, I NOMI DELLO SCANDALO CHE HA TRAVOLTO LA CHIESA CATTOLICA NEL MONDO

LUCA POLLINI

L'espresso
«SCACCO AL PAPA»

The New York Times
«UN PAPA PUÒ DIMETTERSI?»

DER SPIEGEL
«RATZINGER: (IN)FALLIBILE»

The Washington Post
«COME È PIÙ DEL WATERGATE»

Libération
«BISOGNA CAMBIARE PAPA?»

Süddeutsche Zeitung
«LA CHIESA HA NASCOSTO»

IN EDICOLA